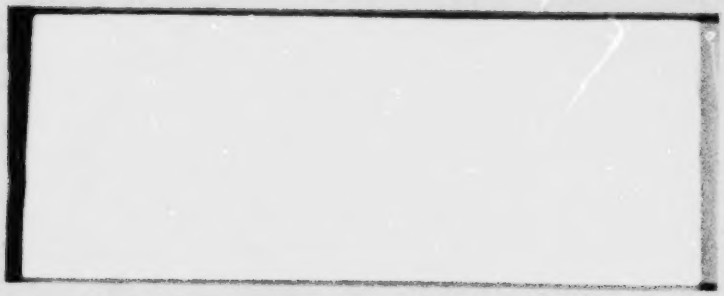
TCS

LIMITED

ENGINEERING RESEARCH



T E S Report Number: C277-2

9-9040

CONTRACT SERIAL NO: OEZ79-05046
REQUISITION NO: 23440-9-9040
CONTRACTOR: TES Limited

A MICROCOMPUTER BASED REAL-TIME DATA ACQUISITION AND DISPLAY SYSTEM

March, 1980

Prepared for

Department Of Energy, Mines And Resources
CANMET

By

T E S Limited

Ottawa, Canada

140 pp

MICROMEDIA

DIRECTORY

	Page No
1. CDOS OPERATING SYSTEM	1
2. REFERENCES	2
3. SYSTEM OVERVIEW	3
3.1 Hardware Description	3
3.2 Software Description	5
3.2.1 DISP.COM	6
3.2.2 COMPAC.COM	7
3.3 Online Calculations	12
4. OPERATION	13
4.1 Power On Start	13
4.2 Cold Start	13
4.3 Using the Keyboard	14
4.4 Preparation for a Run	17
4.5 Starting a Run	17
4.6 Manual Data	18
4.7 Barrel Change	18
4.8 Alarms	18
4.9 Changing Diskettes	19
5. FILE FORMATS	20
5.1 Input Files	20
5.1.1 BIGTAB.DAT	20
5.1.2 SHORT.DAT	22
5.1.3 LONG.DAT	22
5.1.4 KSTS.DAT	22
5.1.5 LOCS.DAT	23
5.1.6 MANDAT 1.DAT	23
5.1.7 MANDAT 2.DAT	23
5.1.8 DISP.DAT	23

DIRECTORY (cont'd)

	Page No.
5.2 Output Files	24
5.2.1 MARKER.DAT	24
5.2.2 PLANT 1.DAT or PLANT 2.DAT	24
6. HARDWARE	27
6.1 4FDC Modifications	27
6.1.1 ROM Modifications	27
6.1.2 Wiring Modifications	27
6.2 TUART Wiring Modifications	28
6.3 Wire Wrap Interface Board	28
6.4 Switch Settings	32
6.4.1 CUP Board	32
6.4.2 4FDC Board	33
6.4.3 64KZ Memory Board	33
6.4.4 TUART Board	34
6.4.5 Cybernex LGR1 Terminals	35
APPENDIX "A" - Source Listings	36

1. CDOS OPERATING SYSTEM

NOTE: It is not within the scope of this manual to describe in detail the operation of the Cromemco CDOS operating system. Many of the steps in preparing to use the software described in this manual require a knowledge of the CDOS operating system. Methods of making back-up copies and other file-transfering operations are not discussed here because they are described in the CDOS manuals.

2. REFERENCES

The following documents supply additional information pertinent to a proper understanding of the contents of this volume:

1. Fluke Model 2240B Instruction Manual
2. PHOIL System Operations Manual
3. PHOIL System Software Manual
4. Cybernex LGR1 Reference Manual
5. Cybernex LGR1 Maintenance Manual
6. Cromemco TUART Instruction Manual
7. Cromemco 64KZ Instruction Manual
8. Cromemco ZPU Instruction Manual
9. Cromemco Text Editor Instruction Manual
10. Cromemco Screen Editor Instruction Manual
11. CDOS User's Manual
12. Screen Editor User's Manual
13. Fortran User's Manual
14. RATFOR User's Manual
15. Software Tools

The last five manuals are included with the "Fortran with RATFOR" software package from Cromemco.

3. SYSTEM OVERVIEW

3.1 Hardware Description

A Cromemco System II, with three 5¼ inch floppy disk drives has replaced the HP9830 calculator as the central processing element of the new system. Because of its greater memory capacity, faster access to data files and its programming language versatility, this new computer allows the operator much faster access to information than the old system.

A number of hardware modifications were made to the system and these are explained in more detail in another section of this manual. A brief description of these modifications follows:

1. The RDOS prom on the 4FDC disk controller board was replaced with one that will allow power failure recovery without operator intervention.
2. A third disk drive was added to the basic 2-drive System II.
3. A modified TUART board was added to the system to control the two video terminals.
4. A wire-wrap interface board was built and added to the system. This board contains the hardware required to interface the System II to the Fluke data logger and, if necessary, to the HP9830.

Two Cybernex LGR-1 video terminals were added to the system. Under control of the System II computer, these terminals display all the channel data and calculation results in three pages of data tables. The operator can access the next page by hitting a single key. Alarms are signalled by a blinking display of the out-of-limit data and a bell ringing each time the reading is updated. In addition to the data

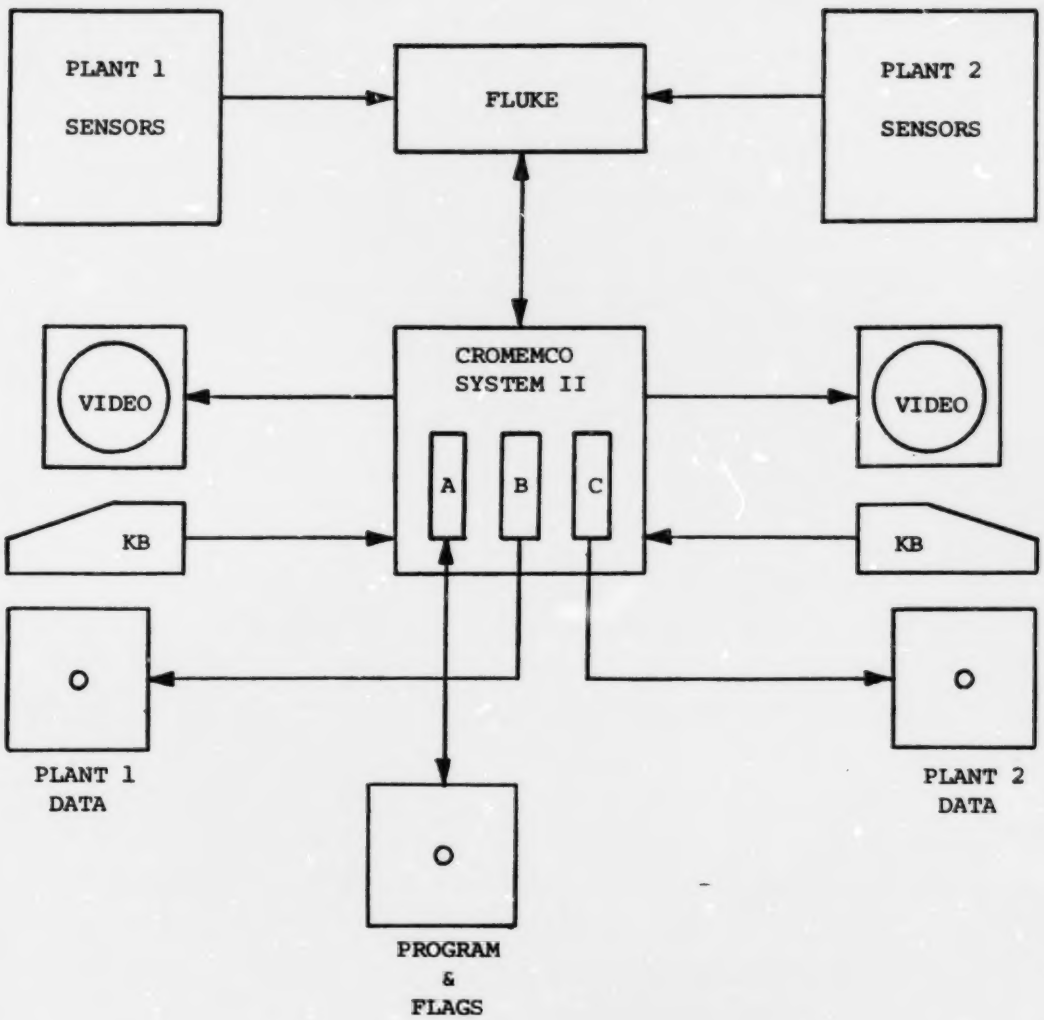


FIGURE 1

SYSTEM BLOCK DIAGRAM

tables, the screen displays time, date, program status and plant status.

During interrupt routines, questions are displayed in the upper left corner of the screen and the answers are echoed to the operator in the upper right corner of the screen. The keyboards operate under interrupt control so the main program can continue to read and display channels while the operator enters his data. Response to the keyboards will be immediate, except while the system is writing to the disks or writing a table to the screen. Both operators can type at the same time without mutual interference except during the short delays mentioned above.

Figure 1 shows a block diagram of the hardware configuration. There are several channels of buffered parallel interface available for future expansion.

3.2 Software Description

The software is supplied on two diskettes. The first diskette, called the DATA LOGGER PROGRAM DISKETTE will "bootstrap" only in the machine located in Building 4. This is a CDOS system diskette and contains a special version of CDOS which will only work on that machine. Also on this diskette are the main program DISP.COM and its data files.

The other diskette, called the COMPAC diskette, is also a CDOS system diskette but it has a more normal version of CDOS and will "boot" up on almost any standard Z2D system. It will not, however, work on the machine in Building 4. This diskette contains the input data files which the engineer can edit to prepare the system for operation. Once all the files are prepared, the program COMPAC.COM can be run to reformat this data into a form which the main program can use efficiently. The output of this program is a 12k file called DISP.DAT. After running COMPAC,

this file can be written to the DATA LOGGER PROGRAM DISKETTE using the XFER command.

The two main programs, DISP.COM and COMPAC.COM, are described below.

3.2.1 DISP.COM

This program has three main functions:

1. To provide the operators of the two pilot plants with quick access to all the various temperature, pressure, weight and flow rate readings associated with their plant.
2. To signal the operator if any of the readings differ from a preset center point by an amount greater than a preset tolerance.
3. To store an hourly record of all the readings and some manually entered data from each plant in such a way that they can be recovered by a computer for later processing.

The previous system, using an HP9830 as the computer, provided the last two functions but was unable to provide immediate access to information due to a lack of display hardware. In the new system, two separate terminals provide the operators with access to information about their own plant without interference from the other plant. The information is displayed in an easily readable fashion with a description of each channel's function accompanying the displayed value. The expected center point and units for each value are also displayed. The value itself is highlighted on the screen for easy separation from the other information displayed.

In addition to the channel data, some status information is displayed at the top of the screen including the time, date, run number and program status.

Unlike the old system which had only one keyboard for both operators, the new system provides each plant with its own keyboard allowing each operator access to the program from his own operating position. These keyboards allow the operators to change center points and tolerances, enter manual data, change diskettes, etc. without disturbing each other. Because the keyboards work under interrupt control, there are only very short delays in response to commands and the program continues to read channels while the operator enters his commands or data.

The Fluke datalogger can read up to 180 channels, numbered Ø - 179. In addition, each plant can have up to 22 calculations which are numbered 180 - 223. Of these 224 channels, 112 may be assigned to each plant. The position where a given channel may appear on the screen or in the hourly record, can also be assigned by the engineer. An AX + B type of conversion may be performed on any channel simply by choosing appropriate values for A and B.

3.2.2 COMPAC.COM

The engineer supplies the above information including the channel description, units, A, B, center, tolerance and position to the program in the form of a table called BIGTAB.DAT. This table is read by a program called COMPAC.COM. This program reads the table and compacts it into a form which the display program can use efficiently. COMPAC.COM also reads four other tables which the engineer must supply.

The first is a simple list of constants used in the online calculations. This table is called KSTS.DAT. The second is a list of channel numbers where the information used in the online calculations may be found. This table is called LOCS.DAT. The third and fourth tables are two lists of channel numbers which determine the order in which the channels will be read by the program. One of these lists is called SHORT.DAT. This is a list of more important channels. The other list is called LONG.DAT and includes less important channels.

In the new system, any reading from the Fluke may be averaged over "N" readings simply by entering "N" into the table contained in the file BIGTAB.DAT.

The channels on the Fluke data logger are numbered Ø - 179. This numbering system was extended to include the calculations. Numbers 180 - 223 can be assigned to various calculations in the online calculation subroutines. Except for the method of arriving at a value for these numbers, they are all treated the same way from the point of view of alarm checking, display on the screen, and hourly storage.

Some of these channels need only be read once each hour while others should be updated as often as possible. It takes about 3 seconds to read and display one channel so if the channels were simply read in numerical order, each channel would be read about every 12 minutes. This is too often for some and not often enough for others.

To overcome this problem, the program reads the next channel to process by alternating between two lists supplied by the operator. One list is short and contains the channels which must be read more often and

the other longer list contains the rest of the channels. A channel may appear more than once in a list and if a channel is not required it may be left out of both lists. Since calculations are numbered just like the Fluke channels, their order of execution can be controlled in the same manner using the two lists.

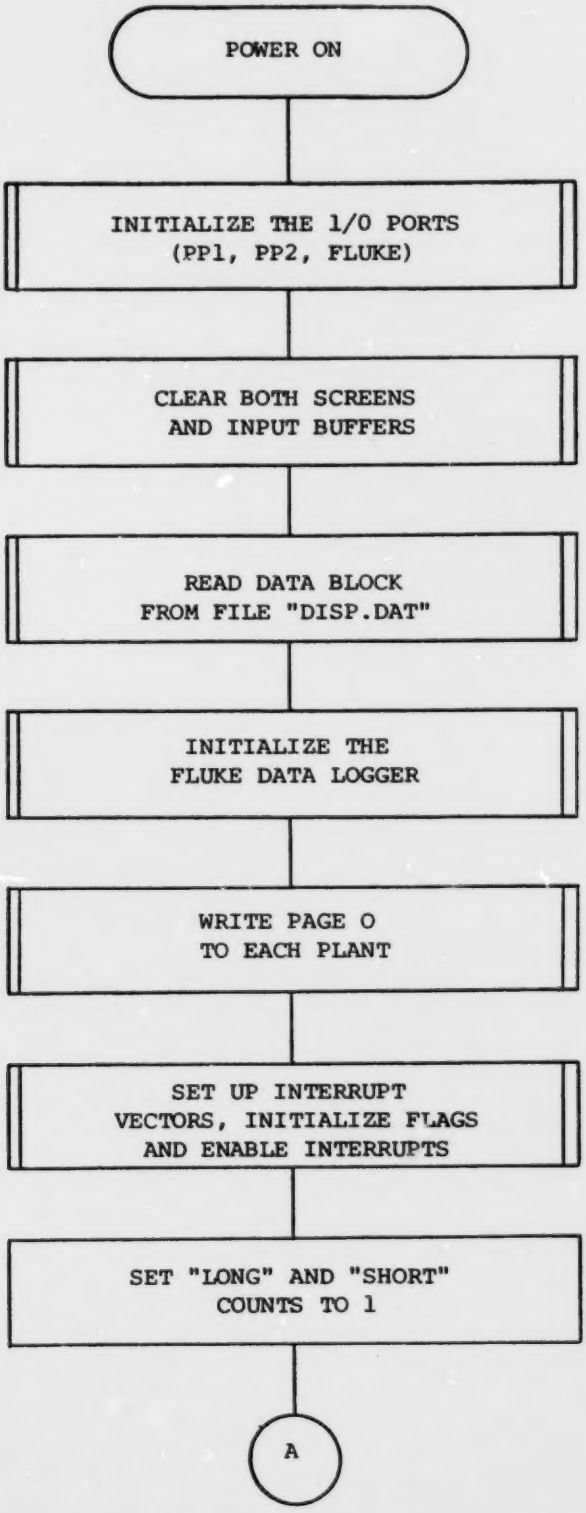


FIGURE 2A

SYSTEM FLOW CHART

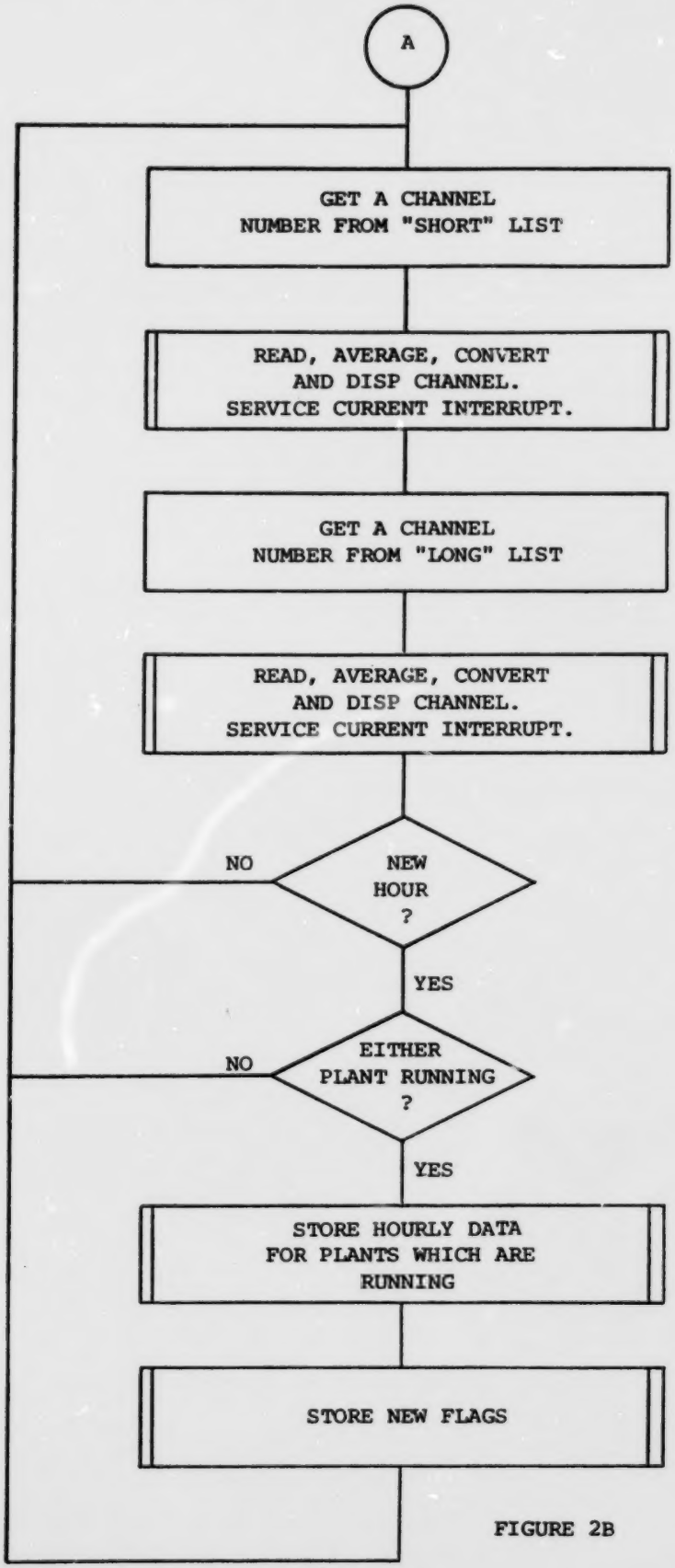

A
GET A CHANNEL NUMBER FROM "SHORT" LIST
READ, AVERAGE, CONVERT AND DISP CHANNEL. SERVICE CURRENT INTERRUPT.
GET A CHANNEL NUMBER FROM "LONG" LIST
READ, AVERAGE, CONVERT AND DISP CHANNEL. SERVICE CURRENT INTERRUPT.
NEW HOUR ?
NO
YES
EITHER PLANT RUNNING ?
NO
YES
STORE HOURLY DATA FOR PLANTS WHICH ARE RUNNING
STORE NEW FLAGS

FIGURE 2B

3.3 Online Calculations

At the time of writing this manual, the online calculations were written to match those documented in the PHOIL system software manual. To date, these routines have not been tested and debugged.

Throughout the rest of the software, anything the engineer may want to change is table driven. However, it is not possible to do this with the online calculations. If a change is required, the source code must be altered and then re-compiled. Provision has been made in the software for expansion of these online calculations and it is expected that changes will be made at frequent intervals.

As a result of the above factors, the online calculation source code will have to be largely self-documenting.

4. OPERATION

4.1 Power On Start

Normally the program will load up and begin execution whenever the power to the computer is turned on. This is called a "warm-start". A "warm-start" assumes that no conditions have changed since the power was turned off.

If the program will not start or "hangs-up" shortly after start-up, a "cold-start" may be required.

4.2 Cold Start

1. Bring the following files up to date using the screen editor:
    a. BIGTAB.DAT
    b. KSTS.DAT
    c. LOCS.DAT
    d. LONG.DAT
    e. SHORT.DAT
2. Run COMPAC.COM and check to be sure DISP.DAT is 12k long.
3. Erase DISP.DAT from the data-logger program disk.
4. XFER DISP.DAT from the COMPAC disk to the program disk.
5. Check the directory of the program disk. It should contain the following files:
    a. CDOS COM
    b. @.COM
    c. DISP.COM
    d. DISP.DAT
    e. MANDAT 1.DAT
    f. MANDAT 2.DAT
    g. START-UP.CMD

6. Place the program disk into drive A of the data logging system and turn on the power. The following sequence of events should take place:
    a. The machine will "bootstrap" load CDOS.COM from the program disk.
    b. CDOS will check for a file called STARTUP.CMD.
    c. @.COM will be loaded with STARTUP.CMD as an input file.
    d. The main program - DISP.COM - will be loaded and will begin execution.
    e. The main program will load DISP.DAT, initialize the Fluke and write page Ø to each terminal.
    f. At this point, the interrupts will be enabled and the operators can start preparing for a run by entering the date, Xchanging the disks, etc.

4.3 Using the Keyboard

Each plant now has its own keyboard, therefore, the computer never asks "which plant?". This information is implied by the keyboard being used.

Normally, the computer is reading and displaying channels and waiting for an interrupt command to be entered at the keyboard. Interrupt commands are all control characters A to Z. A control character is sent by holding down the control key at the lower left of the keyboard, while pressing the appropriate letter simultaneously. If an invalid or out-of-sequence key is pressed, the terminal will respond with a bell.

When a valid command is received, the computer will finish the channel it is reading and then respond with a question in the upper left corner of the screen.

There are three types of answers that the computer will accept. Depending on the question asked, the computer will accept only answers conforming to one of the types described below:

1. Numbers - Only the digits Ø - 9, a decimal point or a minus sign will be accepted. A minus sign can only be entered in the first character position and only one decimal point will be accepted. If an error is made, the delete key will erase one character each time it is pressed. Invalid characters will not be accepted and the terminal bell will ring.
2. Yes or No - In this mode, the only character that the computer will recognize is Control Y. This is interpreted as a "yes" answer. Any other charcter will be interpreted as a "no" answer.
3. Word - In this rarely-used mode, the computer will accept any printable character.

In any of the above modes, the characters accepted will be displayed in the upper right hand corner of the screen as soon as the computer can gain access to the screen. Note that the delay is not in accepting the character but in displaying it. It would be quite impossible to type faster than the machine can accept the characters.

Always check the value you have entered to make sure it is correct, then hit "carriage return" to tell the computer you have finished typing in the answer. A list of interrupt commands is shown in the following table.

CONTROL CHARACTER	FUNCTION
A	Auto Initialize
B	Barrel Change
C	
D	Day & Year
E	Edit Center Point
F	
G	
H	
I	
J	
K	
L	
M	
N	
O	Other Page
P	
Q	Quit Run
R	Run Start-up
S	
T	Tolerance Edit
U	Uncover Incomplete Files
V	
W	Write Manual Data
X	Xchange Diskette
Y	Yes Answer
Z	Zap Program

TABLE 1

INTERRUPT COMMANDS

4.4 Preparation for a Run

Before starting a run, the following things should be checked:

1. Check that the date, year, and time are correct. If time is wrong, set the clock on the Fluke data logger as described in the Fluke manual. If the day or year is wrong, use Control D to correct them.
2. Check that the center values and tolerances are set for the expected values during the run. To edit these, use Control E or Control T. By using Control A, all channels with a negative tolerance on the one plant will have their center values set to the present reading. This feature can save the operator from having to edit the center values of the less important channels.
3. Make sure that there is a data diskette in the drive for the plant which is about to start a run. If the diskette has been used for a previous run, check the directory by using Control U to make sure there is plenty of space available. If the diskette is blank or you are not sure of its status, change diskettes using Control X.

4.5 Starting a Run

To start the run, use Control R. The machine will ask the appropriate questions and then store a "start-up" file. Manual data collected at the time of the start-up can be written to this "start-up" file by using the Control W command.

The computer will now begin storing hourly data until the run is terminated by the operator.

4.6 Manual Data

Up to 32 manual data points can be entered for each record in the data file. A file of questions is prepared by the operator using the screen editor and stored on the program disk. These files are called MANDAT 1.DAT and MANDAT 2.DAT. For each question, a real-number answer is stored in the order in which the questions were asked.

Manual data can be entered for any day and hour which has already been stored on the data diskette. The Control U command will type out a directory of these files. The files which have not had manual data written into them will be highlighted. If errors are made in the entry of manual data, the data may be re-written to the record for any day and hour, any number of times. The status displayed by the Control U command for each hourly record will represent the number of times the record has had manual data entered plus one for the automatic data.

4.7 Barrel Change

During a run, the light-end and heavy-end barrels will periodically fill up and have to be changed. When one of these barrels is changed, use the Control B command. The computer will collect the data required to keep track of the products.

4.8 Alarms

Any time the system is reading channels and displaying them, it will check for alarm conditions. When an alarm condition is encountered, the computer will blink the out-of-limit value on the screen and ring

the bell on the terminal each time the channel is displayed. To suppress an alarm, either the center value or the tolerance value for that channel can be changed. On less important channels, the Control A command will usually stop the alarm condition.

4.9 Changing Diskettes

Each data diskette can store 5 days' data on each side. In addition to the hourly data, start-up and shut-down files also occupy space on the diskette. If a large number of these extra records are written, the diskette will be full before the 5 day limit. The best way to judge when a diskette should be changed is by using the Control U command. If the diskette is nearly full, the entire screen of the terminal will be filled with directory information and the "change diskette" warning message will be displayed accompanied by a long ringing of the terminal bell.

To change diskettes, first get a blank diskette which has been initialized using the CDOS INIT.COM program, then use the Control X command and follow the instructions given by the program.

5. FILE FORMATS

5.1 Input Files

5.1.1 BIGTAB.DAT

Each line of this table represents one channel or calculation number. The information contained in this table and its position on the line are shown in the editing guide. The following is a brief outline of the restrictions on each of the entries.

Channel Number - These are 3 digit numbers in the range of 0 through 223. Each channel number must appear only once in the table.

Plant - This number assigns the channel to either of the two plants. The number must be either 1 or 2. Only 112 channels may be assigned to each plant but any channel may be assigned to either plant.

Disk - Any 6 letter description may be used here.

Units - Any 2 letter unit code may be used.

Priority - Not implemented at present - zero

N - This is the number of samples to be averaged while reading a channel. It can be a 3 digit number but the lowest number that gives good results should be used. For most channels, 1 will be used.

Page - This is the page number where the results will be displayed. It must be 0, 1 or 2.

X Position - This is the column in which the data will be displayed. It must be 1, 2 or 3.

Y Position - This is the row in the display table where the result will appear. It must be between 1 and 17 inclusive.

EDITING GUIDE FOR BIGTAB.DAT

ENTRY KEY

CHANNEL	PLANT	DESC	UNITS		N PAGE	X POS	Y POS	SECT	WORD	"A"	"B"	CENTER	TOLERANCE
0	1	CHN000	C	0	1	0	1	1	1	1.000	0.000	500.000	-500.000
1	1	CHN001	C	0	1	0	1	2	1	2	1.000	0.000	500.000	-500.000
2	1	CHN002	C	0	1	0	1	3	1	3	1.000	0.000	500.000	-500.000
221	2	RES221	C	0	1	2	1	8	4	14	1.000	0.000	500.000	-500.000
222	2	RES222	C	0	1	2	1	9	4	15	1.000	0.000	500.000	-500.000
223	2	RES223	C	0	1	2	1	10	4	16	1.000	0.000	500.000	-500.000

PRIORITY

COLUMN NUMBER GUIDE

0	1	2	3	4	5	6	7
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789							
0 1 CHN000 C 0 1 0 1 1 1 1				1.000	0.000	500.000	-500.000
1 1 CHN001 C 0 1 0 1 2 1 2				1.000	0.000	500.000	-500.000
2 1 CHN002 C 0 1 0 1 3 1 3				1.000	0.000	500.000	-500.000
3 1 CHN003 C 0 1 0 1 4 1 4				1.000	0.000	500.000	-500.000

Sector - This is the number of the sector where the result will be stored in the hourly record. It must be between 1 and 4.

Word - This number positions the data in the sector. Each of the first 3 sectors contain 32 words and the 4th contains 16 words.

"A" - After a channel is read and averaged, it is multiplied by this real number. For channels not requiring a conversion factor, this number should be 1.0.

"B" - After the value is multiplied by "A", "B" is added to the result. For channels not requiring a conversion, "B" should be 0.0.

Center - This is set to the expected value for the channel. If the value of the channel differs from this center value by more than the tolerance, an alarm condition will result.

Tolerance - This number sets the ± tolerance for the alarm checking. If a negative number is entered here, the channel will auto initialize when the operator uses the Control A command.

5.1.2 SHORT.DAT

Each line in this file contains one channel number. The file is a list of up to 100 important channels.

5.1.3 LONG.DAT

This file is similar to SHORT.DAT except that it contains a list of less important channels up to a maximum of 500 entries.

5.1.4 KSTS.DAT

This file contains a list of constants used in the online calculations. At present, this file can contain up to 10 entries.

5.1.5 LOCS.DAT

This file contains a table of channel numbers. During an online calculation, the values of variables used in the calculation are read from various channels. From time to time, the channel number where a particular thermocouple or other sensor is connected may be changed. This table eliminates the need to then change the formula used in the calculation. Instead, the entry in the table is changed. The format of this table changes as the online calculations change, so comment lines are included in the table to explain the meaning of the various entries. A comment line must start with a "#" sign.

5.1.6 MANDAT 1.DAT

Each line of this file contains a question which the manual data subroutine will ask each time it is called by Plant 1. As each question in the file is asked, the answer will be stored as a real number. Up to 32 questions may be included in the file. When all the questions are answered, the subroutine will store these answers in Sector 5 of the hourly data record.

5.1.7 MANDAT 2.DAT

This file is similar to MANDAT 1.DAT but contains the questions for Plant 2. Both MANDAT 1.DAT and MANDAT 2.DAT must be present on the DATA LOGGER PROGRAM DISK.

5.1.8 DISP.DAT

Although this is an output file for the COMPAC.COM program, it is included here as an input file because it is the main input file for

the DISP.COM program. This file contains all the information from the tables described above as well as the system flags. Each time these flags change, the DISP.COM program writes the new flags to this file.

DISP.DAT is a binary file and can only be read by an unformatted read in a Fortran program.

5.2 Output Files

Each plant has its own disk drive. The data diskettes will contain only two files. Both of these files are actually read/write files but, because they contain the output data from the program, they are classed here as output files.

5.2.1 MARKER.DAT

At present, this file only contains the word "used" if the diskette contains data. Since files have a minimum length of 1k, this is very inefficient. The remainder of this file will be used for other things when the program is expanded. This file is created by the Control X interrupt routine. This routine first reads the file and, if the file is not empty, it signals the operator that the diskette is not empty.

5.2.2 PLANT 1.DAT or PLANT 2.DAT

These are the hourly data files for the two plants. The first 1k of the file contains a directory. The rest of the file contains the hourly records which occupy 5 sectors each. The first 3 sectors contain 32 numbers each. The fourth sector is divided in half. The

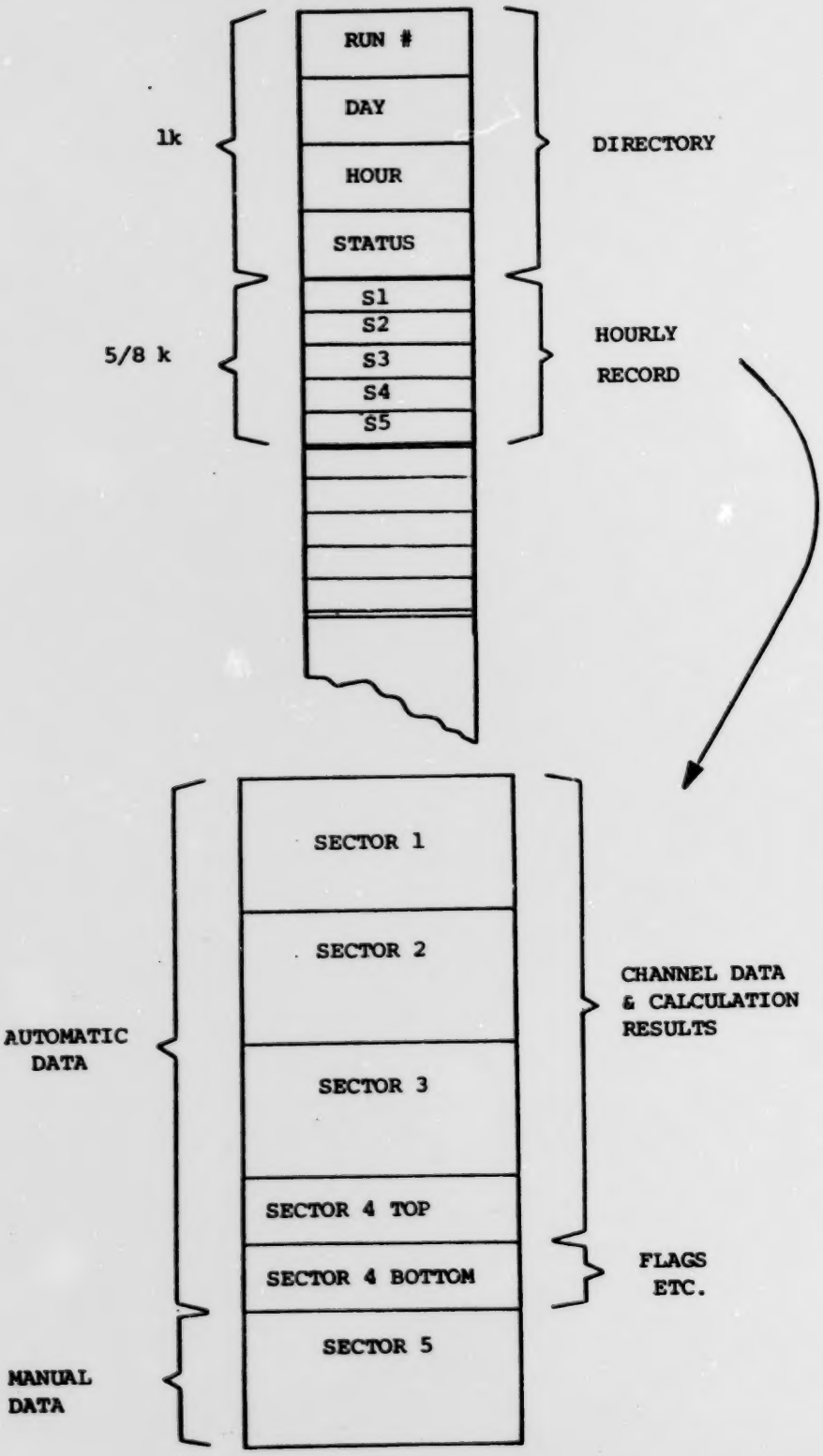
top half contains 16 numbers and the bottom half contains other data which must be stored each hour. The fifth sector contains up to 32 numbers collected by the manual data entry subroutine. Each time data is written to an hourly record, the directory is updated.

Each directory entry contains the following information:

1. Hourly data status;
   0 = empty
   1 = automatic data only
   2 or greater = complete
2. Hour of the day
   24 = start-up file
   25 = shut-down file
3. Day of the year
4. Run Number

The directory can have up to 128 entries. This is enough for 5 days data and 8 start-up or shut-down files.

The following diagrams show the layout of the data files.


RUN #
DAY
HOUR
STATUS
S1
S2
S3
S4
S5
1k
5/8 k
DIRECTORY
HOURLY
RECORD
SECTOR 1
SECTOR 2
SECTOR 3
SECTOR 4 TOP
SECTOR 4 BOTTOM
SECTOR 5
AUTOMATIC
DATA
MANUAL
DATA
CHANNEL DATA
& CALCULATION
RESULTS
FLAGS
ETC.

6. HARDWARE

6.1 4FDC Modifications

6.1.1 ROM Modifications

The ROM on the 4FDC disk controller board contains the RDOS monitor and a "bootstrap" program. The "bootstrap" program initializes the hardware and reads the CDOS monitor from the diskette in drive A. In a normal System II, the "bootstrap" program waits for the operator to enter a few carriage return characters from the terminal device so that the baud rate of the UART can be set to the same rate as the terminal.

This means that the System II cannot recover from a power failure without operator intervention.

In a normal System II, the terminal device is assigned to the UART on the 4FDC board. Our system required interfacing to two terminal devices, both of which had to operate under interrupt control. The two terminals were assigned to the TUART board and the Fluke data logger was assigned to the UART on the 4FDC board.

To make the changes outlined above, the ROM on the 4FDC board was replaced with a 2708 EPROM with a modified program.

6.1.2 Wiring Modifications

One jumper was added to the 4FDC board; it runs from Pin 5 of the TMS5501 (IC12) to the unused Pin 26 of the cable stripe J4.

This jumper carries the "buffered received data" to the "CTS" flip-flop on the interface board.

6.2 TUART Wiring Modifications

There are some unused RS232 line drivers and receivers on the TUART board. The following modifications were made so that these drivers and receivers could be used by the handshaking hardware between the computer and the Fluke data logger.

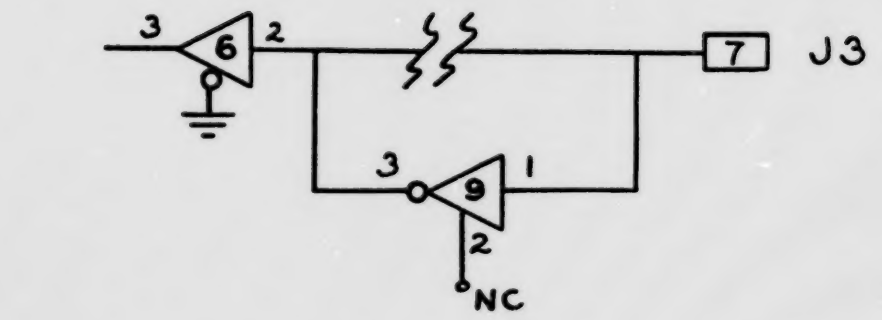
1. The connector between Pin 2 of IC6 and Pin 7 of cable stripe J3 was cut.
2. A jumper was added between Pin 7 of cable stripe J3 and Pin 1 of IC9.
3. A jumper was added between Pin 26 of cable stripe J2 and Pin 4 and 5 of IC11.
4. A jumper was added between Pin 3 of IC9 and Pin 2 of IC6.
5. A jumper was added between Pin 6 of IC11 and Pin 26 of cable stripe J3.

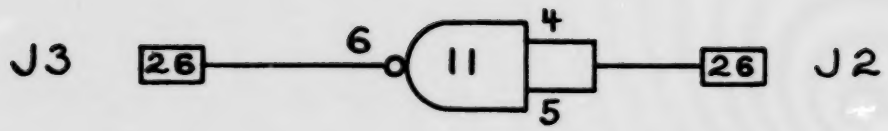
6.3 Wire Wrap Interface Board

The interface board is not connected to the bus, but is mounted in a bus slot. This board serves the following functions:

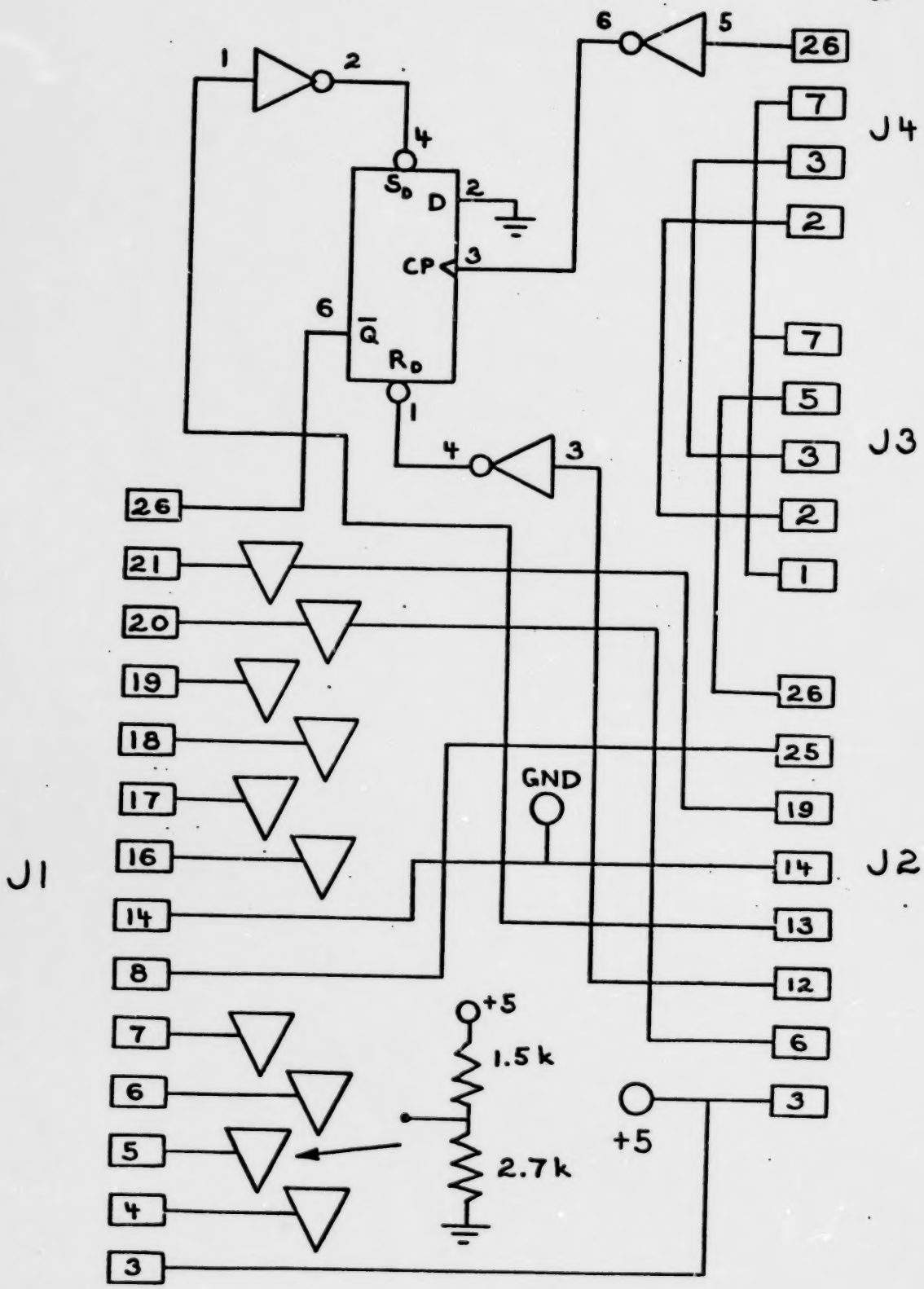
1. It provides a method of transferring signals between the many interface cables in an orderly and reliable manner.
2. It provides the required circuitry to send a "clear-to-send" handshake signal from the computer to the Fluke data logger.
3. It provides the resistive termination network required by the HP11202A parallel interface.

A wiring diagram of this board is shown in the following diagrams.

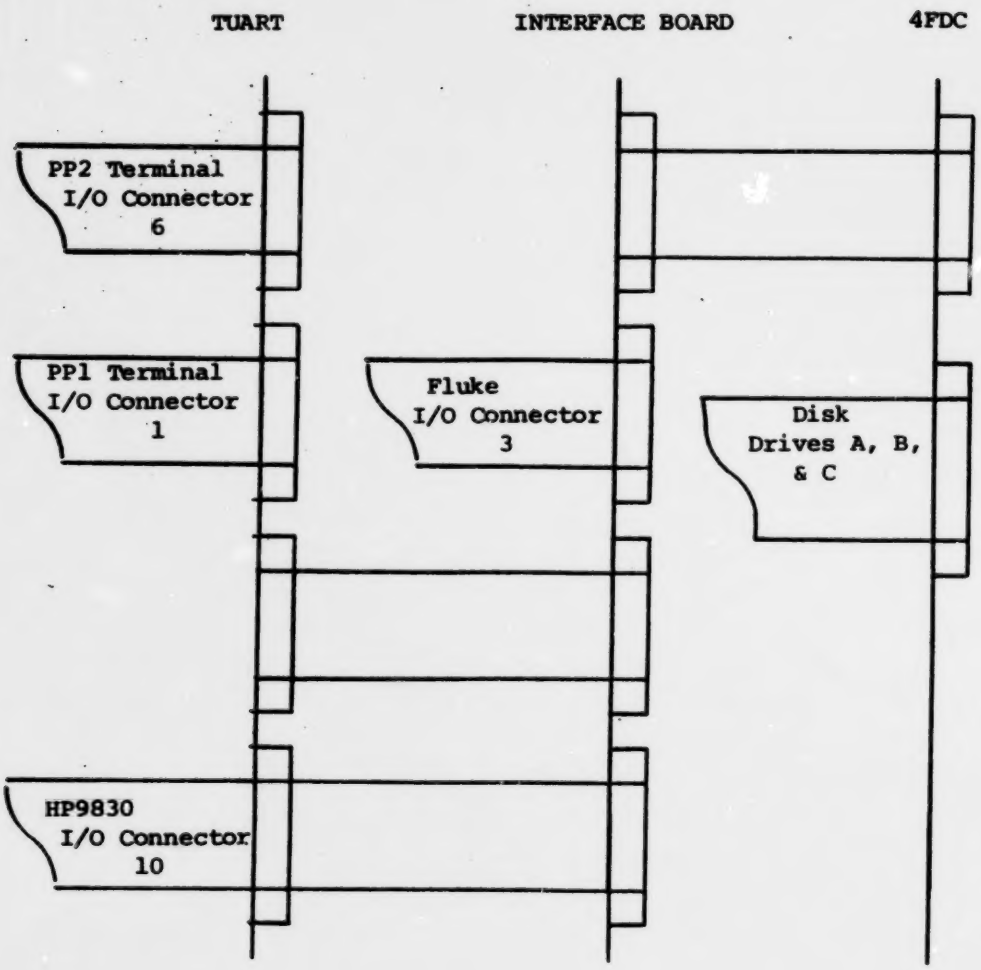




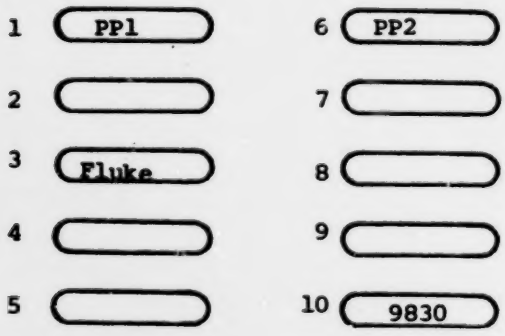
TUART WIRING MODIFICATIONS



INTERFACE BOARD SCHEMATIC



I/O CABLE LOCATIONS
(Front View of Bus Slots)

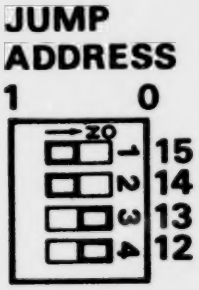


I/O CONNECTORS
(Rear View of System II)

6.4 Switch Settings

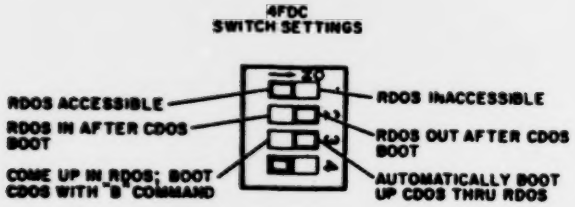
Each device or circuit board in the system has its own set of switches. For this system to work properly, each switch must be set properly. The purpose of each switch is explained in the appropriate manual for the device or board. This section of the manual will show the proper setting for each switch without explanation.

6.4.1 CPU Board

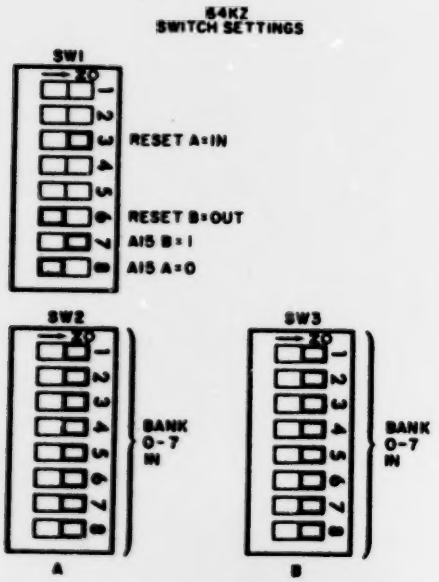


The red toggle switch labeled "2" and "4" should be set to "4".

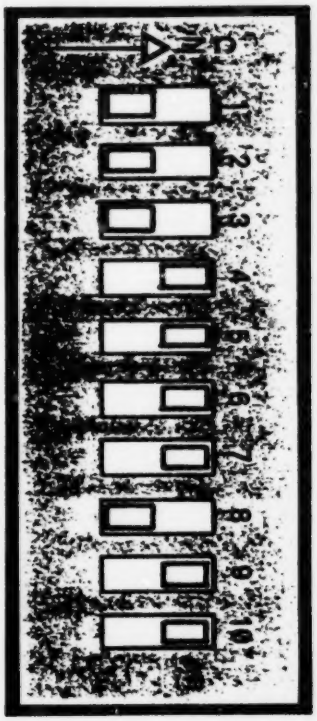
6.4.2 4FDC Board


4FDC
SWITCH SETTINGS
RDOS ACCESSIBLE
RDOS IN AFTER CDOS BOOT
COME UP IN RDOS; BOOT CDOS WITH "B" COMMAND
RDOS INACCESSIBLE
RDOS OUT AFTER CDOS BOOT
AUTOMATICALLY BOOT UP CDOS THRU RDOS

6.4.3 64KZ Memory Board



6.4.4 TUART Board



Caution: Base addresses 00H and 30H are used by Cromemco's 4FDC Floppy Disc Controller; 40H is used by Cromemco memory boards with BANK SELECT; and 50H is used by Cromemco's PRI Printer Interface.

6.4.5 Cybernex LGR1 Terminals

LOCAL/REMOTE	REMOTE
FULL/HALF DUPLEX	FULL
PAGE/ROLL	PAGE
PARITY	NONE
BAUD RATE	9600
CAPS LOCK	LOCKED

APPENDIX "A"

SOURCE LISTINGS

A. SOURCE LISTINGS

A.1 INDEX

A.2 I/O DEVICE DRIVERS FOR CDOS

A.3 NLUN.Z80 ASSEMBLY LANGUAGE SUBROUTINES

- TABLE OF EQUATES
- LOGICAL UNIT NUMBER TABLE -
- DISPATCH TABLE TERM PP1 -
- DISPATCH TABLE TERM PP2 -
  - REWIND, BACKSPACE & ENDFILE
  - BINARY READ & WRITE
  - FORMATTED READ
  - FORMATTED WRITE PP1
  - FORMATTED WRITE PP2
  - DEVICE DRIVER PP1
  - DEVICE DRIVER PP2
- CURSAD
- CONV
- FLUKE DISPATCH TABLE
  - FORMATTED READ
  - FORMATTED WRITE FLUKE
  - FLUKE INPUT DRIVERS
  - FLUKE OUTPUT DRIVER
- TUART
- STOP
- ECHO1 & ECHO2
- MODSET
- ISTAT
- ICR
- WRTBAF

WRTBUF
SETTMB
CLRBUF
GETBUF
TERMIN
A.4 COMMON.RFR
A.5 COMBLOK.RFR
A.6 DISP.RFR
MAIN
A.7 SUBS.RFR
DOCHN
CLEAR
DUMPF
GETBLK
FILL
DISP
PROGST
LEFTJ
A.8 STORE.RFR
HRLY
STORE
A.9 FLUKE.RFR
FINIT
SETCLK
COMAND
NDSCAN
AVCHN
ZAPLF
A.10 DISPAT.RFR
DISPAT
DUMMY
OTHER
AUTO I

BARREL

CLRTOP

RIGHTJ

DAYIN

WRTMAN

XCHNGD

ZAP

GDIR

WDIR

EDIT

UNCOV

STRTUP

SHTDWN

A.11 ONLINE.RFR

ONLINE

FEED

HTC

DTRC

GASR

A.12 COMPAC.RFR

COMPAC

LDBIG

LDBATO

LDKSTS

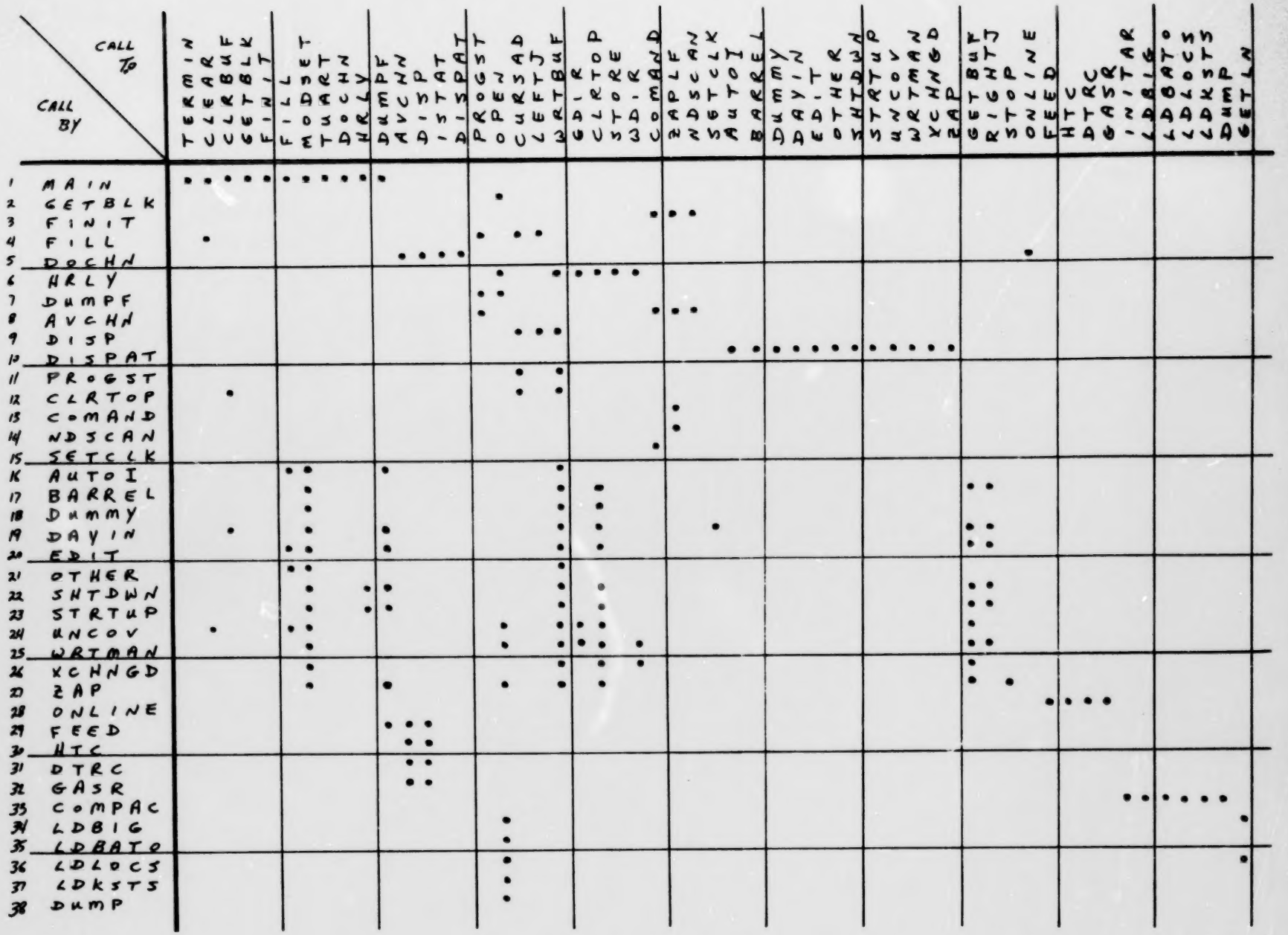
DUMP

LDLOCS

INITAR

INDAT

GETLN



Cross Reference of Subroutine Calls

A.2
I/O DEVICE DRIVERS
FOR SPECIAL CDOS

```
            TITLE   CANMET I/O DEVICE DRIVERS FOR CDOS
            REM
            REM     THIS VERSION FOR CANMET
            REM
            REM
            LIST    NOCOND
            REM
;
NULLS       EQU     0               ; NUMBER OF NULLS TRANSMITTED AFTER LINE FEED
;
;
;
;   ASCII CHARACTERS
;
CTRLB       EQU     2               ; ASCII CONTROL-B CHARACTER
BACK        EQU     8               ; ASCII BACK SPACE
LF          EQU     0AH             ; ASCII LINE FEED
VT          EQU     0BH             ; ASCII VERTICAL TAB
FF          EQU     0CH             ; ASCII FORM FEED
CTRLN       EQU     0EH             ; ASCII CONTROL-N CHARACTER
CTRLO       EQU     0FH             ; ASCII CONTROL-O CHARACTER
CTRLP       EQU     10H             ; ASCII CONTROL-P CHARACTER
CTRLZ       EQU     1AH             ; ASCII CONTROL-Z CHARACTER
ESC         EQU     1BH             ; ASCII ESCAPE CHARACTER
CTRL. UP    EQU     1EH             ; ASCII CONTROL-^ CHARACTER
            TITLE   I/O DEVICE PORT ASSIGNMENTS, STATUS BITS, AND BAUD RATES
;
;
;   I/O DEVICE PORT ASSIGNMENTS AND STATUS BITS
;
BAUD        EQU     10H
COMMND      EQU     12H
CSTATP      EQU     10H             ; CONSOLE STATUS PORT (INPUT)
CDATA       EQU     CSTATP+1        ; CONSOLE DATA PORT (INPUT/OUTPUT)
CRDA        EQU     40H             ; CONSOLE RECEIVER-DATE-AVAILABLE BIT
CTBE        EQU     80H             ; CONSOLE TRANSMITTER-BUFFER-EMPTY BIT
;
RSTATP      EQU     0               ; SERIAL READER STATUS PORT (INPUT)
RCMD        EQU     RSTATP+2
RBAUD       EQU     RSTATP          ; SERIAL READER BAUD RATE PORT (OUTPUT)
RDATA       EQU     RSTATP+1        ; SERIAL READER DATA PORT (INPUT)
RRDA        EQU     40H             ; SERIAL READER RDA BIT
;
PSTATP      EQU     0               ; SERIAL PUNCH STATUS PORT (INPUT)
PBAUD       EQU     PSTATP          ; SERIAL PUNCH BAUD RATE PORT (OUTPUT)
PDATA       EQU     PSTATP+1        ; SERIAL PUNCH DATA PORT (OUTPUT)
PTBE        EQU     80H             ; SERIAL PUNCH TBE BIT
;
LSTATP      EQU     24H             ; LIST DEVICE STATUS PORT (INPUT)
LDATA       EQU     14H             ; LIST DEVICE DATA PORT (OUTPUT)
LRTP        EQU     4               ; LIST DEVICE READY-TO-PRINT BIT
CMDP        EQU     LSTATP          ; LIST COMMAND PORT
;
;
;
```

```
;   I/O DEVICE BAUD RATE ASSIGNMENT TABLE FOR TUART
;
;       01H =  110 BAUD / 2 STOP BITS
;       82H =  150 BAUD / 1 STOP BIT
;       84H =  300 BAUD / 1 STOP BIT
;       88H = 1200 BAUD / 1 STOP BIT
;       90H = 2400 BAUD / 1 STOP BIT
;       A0H = 4800 BAUD / 1 STOP BIT
;       C0H = 9600 BAUD / 1 STOP BIT
;   (REFER TO TUART MANUAL FOR OTHER RATE OR STOP BIT CONFIGURATIONS)
;
;   THE BAUD RATES BELOW WERE CHOSEN FROM THE TABLE ABOVE
;
RDR.BD.RT EQU   88H     ; BAUD RATE OF SERIAL READER
PUN.BD.RT EQU   84H     ; BAUD RATE OF SERIAL PUNCH
SER.BD.RT EQU   88H     ; BAUD RATE OF SERIAL PRINTER
        TITLE   DEVICE DRIVER ADDRESS TABLE
;
;
;   THE FOLLOWING IS A TABLE OF ADDRESSES NEEDED BY CDOS TO
;   LOCATE THE STARTING LOCATIONS OF EACH OF THE I/O DEVICE
;   ROUTINES.   THE ADDRESS VALUES ARE FILLED IN BY CDOSGEN;
;   THEREFORE, THIS TABLE SHOULD NOT BE REMOVED FROM THE DRIVERS.
;
CONSOLE: DW     CINIT           ; CONSOLE INITIALIZE
         DW     CSTAT           ; CONSOLE INPUT-STATUS
         DW     CIN             ; CONSOLE INPUT A BYTE
         DW     CRDY            ; CONSOLE OUTPUT-READY
         DW     COUT            ; CONSOLE OUTPUT A BYTE
         DW     CSET            ; CONSOLE SET SPECIAL COMMAND
;
READER:  DW     RINIT           ; READER INITIALIZE
         DW     RSTAT           ; READER INPUT-STATUS
         DW     RIN             ; READER INPUT A BYTE
;
PUNCH:   DW     PINIT           ; PUNCH INITIALIZE
         DW     PRDY            ; PUNCH OUTPUT-READY
         DW     POUT            ; PUNCH OUTPUT A BYTE
;
PRINTER: DW     LINIT           ; LIST INITIALIZE
         DW     LRDY            ; LIST OUTPUT-READY
         DW     LOUT            ; LIST OUTPUT A BYTE
;
CLOCK:   DW     STRTCLK         ; START CLOCK
         DW     READCLK         ; READ CLOCK
YEAR:    DB     0               ; YEAR (-1900) STORAGE
MON:     DB     0               ; MONTH STORAGE
DATE:    DB     0               ; DATE STORAGE
HOUR:    DB     0               ; HOUR STORAGE
MIN:     DB     0               ; MINUTE STORAGE
SEC:     DB     0               ; SECOND STORAGE
```

```
          TITLE   FUNCTION KEY ADDRESS TABLE
;
;
;   THE FOLLOWING IS A TABLE OF ADDRESSES NEEDED BY CDOS TO
;   LOCATE THE PRE-PROGRAMMED VALUE OF EACH OF THE FUNCTION
;   KEYS.  THE ADDRESS VALUES ARE FILLED IN BY CDOSGEN; THEREFORE,
;   THIS TABLE SHOULD NOT BE REMOVED FROM THE DRIVERS SOURCE.
;
FUNTBL: DW      0       ; FUNCTION KEY F1
        DW      0       ; FUNCTION KEY F2
        DW      0       ; FUNCTION KEY F3
        DW      0       ; FUNCTION KEY F4
        DW      0       ; FUNCTION KEY F5
        DW      0       ; FUNCTION KEY F6
        DW      0       ; FUNCTION KEY F7
        DW      0       ; FUNCTION KEY F8
        DW      0       ; FUNCTION KEY F9
        DW      0       ; FUNCTION KEY F10
        DW      0       ; FUNCTION KEY F11
        DW      0       ; FUNCTION KEY F12
        DW      0       ; FUNCTION KEY F13
        DW      0       ; FUNCTION KEY F14
        DW      0       ; FUNCTION KEY F15
        DW      0       ; FUNCTION KEY F16
        TITLE   CONSOLE ROUTINES
;
;
;   CONSOLE INITIALIZATION ROUTINE
;
CINIT:  LD      A,9
        OUT     COMMND,A
        LD      A,0C0H
        OUT     BAUD,A
        RET                             ;BAUD RATE=9600
;
;
;   GET CONSOLE INPUT STATUS
;   UPON EXIT:  A = -1 (FFH) AND Z-FLAG IS RESET IF CHAR.  IS READY
;               A = 0 AND Z-FLAG IS SET IF CHARACTER IS NOT READY
;
CSTAT:  IN      A,CSTATP        ; GET CONSOLE-IN STATUS
        AND     CRDA            ; CHECK CONSOLE RDA FLAG
        RET     Z               ; CHARACTER NOT READY
        LD      A,-1            ; CHARACTER READY
        RET
;
;
;   CONSOLE INPUT ROUTINE
;   UPON EXIT:  A-REG.  CONTAINS THE CHARACTER READ
;
CIN:    CALL    CSTAT           ; GET CONSOLE-IN STATUS
        JR      Z,CIN           ; LOOP AND WAIT FOR CHARACTER
        IN      A,CDATA         ; READ THE CHARACTER
        AND     7FH             ; STRIP OFF PARITY BIT
        RET

;
```

```
;
;   GET CONSOLE OUTPUT STATUS
;   UPON EXIT:    A = -1 (FFH) AND Z-FLAG IS RESET IF READY FOR CHAR.
;                 A = 0 AND Z-FLAG IS SET IF NOT READY FOR CHARACTER
;
CRDY:    IN       A,CSTATP          ; GET CONSOLE-OUT STATUS
         AND      CTBE              ; CHECK CONSOLE TBE FLAG
         RET      Z                 ; CONSOLE NOT READY FOR CHARACTER
         LD       A,-1              ; CONSOLE READY FOR CHARACTER
         RET
;
;
;   CONSOLE OUTPUT ROUTINE
;   UPON ENTRY:   A-REG. CONTAINS THE CHARACTER TO BE OUTPUT
;
COUT:    PUSH     AF                ; SAVE CHARACTER FOR A MOMENT
COUT30:  CALL     CRDY              ; GET CONSOLE-OUT STATUS
         JR       Z,COUT30          ; ZERO MEANS CONSOLE BUSY
         POP      AF                ; RESTORE CHARACTER
         OUT      CDATA,A           ; OUTPUT THE CHARACTER
       IF  NULLS EQ 0
         RET
       ENDIF
       IF  NULLS NE 0
         CP       LF                ; CHECK FOR END OF LINE
         RET      NZ                ; RETURN IF NOT LINE FEED CHAR.
         LD       A,NULLS+1         ; IF LF, GET NUMBER OF NULLS
COUT50:  DEC      A                 ; CHECK FOR 0 NULLS AT TOP OF LOOP
         RET      Z                 ; RETURN IF ALL NULLS OUTPUT
         PUSH     AF                ; SAVE NULLS COUNTER
         SUB      A                 ; PRINT A SINGLE NULL
         CALL     COUT              ;   CHARACTER (RECURSIVE)
         POP      AF                ; RESTORE NULLS COUNTER
         JR       COUT50            ; LOOP TO PRINT NEXT NULL
       ENDIF
         EJECT
;
;
CSET:    RET
         TITLE    PAPER TAPE OR CARD READER ROUTINES
;
;
;   READER INITIALIZATION ROUTINE
;
RINIT:   LD       A,9
         OUT      RCMD,A
         LD       A,8H
         OUT      RBAUD,A
         LD       A,0
         OUT      CMDP,A
         RET                        ;BAUD RATE=1200
;
;
;   GET READER INPUT STATUS
;   UPON EXIT:    A = -1 (FFH) AND Z-FLAG IS RESET IF CHAR. IS READY
;                 A = 0 AND Z-FLAG IS SET IF CHARACTER IS NOT READY
;
RSTAT:   IN       A,RSTATP
         AND      RRDA
         RET      Z
         LD       A,-1
         RET
;
```

```
;
;   READER INPUT ROUTINE
;   UPON EXIT:    A-REG.  CONTAINS THE CHARACTER READ
;
RIN:      LD        A,1
          OUT       CMDP,A
          LD        A,0
          OUT       CMDP,A
RIN30:    CALL      RSTAT
          JR        Z,RIN30
          IN        A,RDATA
          AND       7FH
          RET
;
          TITLE     PAPER TAPE PUNCH ROUTINES
;
;
;   PUNCH INITIALIZATION ROUTINE
;
PINIT:    RET
;
;
;   GET PUNCH OUTPUT STATUS
;   UPON EXIT:    A = -1 (FFH) AND Z-FLAG IS RESET IF READY FOR CHAR.
;                 A = 0 AND Z-FLAG IS SET IF NOT READY FOR CHARACTER
;
PRDY:     IN        A,PSTATP
          AND       PTBE
          RET       Z
          LD        A,-1
          RET
;
;
;   PUNCH OUTPUT ROUTINE
;   UPON ENTRY:   A-REG.  CONTAINS THE CHARACTER TO BE OUTPUT
;
POUT:     PUSH      AF
POUT30:   CALL      PRDY
          JR        Z,POUT30
          POP       AF
          OUT       PDATA,A
          RET
;
          TITLE     LIST DEVICE ROUTINES
;
LINIT:    LD        A,9
          OUT       COMMND+10H,A
          LD        A,0C0H
          OUT       BAUD+10H,A
          RET                         ;BAUD RATE=9600
;
;
;
;   GET CONSOLE OUTPUT STATUS
;   UPON EXIT:    A = -1 (FFH) AND Z-FLAG IS RESET IF READY FOR CHAR.
;                 A = 0 AND Z-FLAG IS SET IF NOT READY FOR CHARACTER
;
LRDY:     IN        A,LSTATP
          CPL
          AND       2
          RET       Z                 ; CONSOLE NOT READY FOR CHARACTER
          LD        A,-1              ; CONSOLE READY FOR CHARACTER
          RET
```

```
;
;   CONSOLE OUTPUT ROUTINE
;   UPON ENTRY:   A-REG.  CONTAINS THE CHARACTER TO BE OUTPUT
;
LOUT:    PUSH     AF                  ; SAVE CHARACTER FOR A MOMENT
LOUT30:  CALL     LRDY                ; GET CONSOLE-OUT STATUS
         JR       Z,LOUT30            ; ZERO MEANS CONSOLE BUSY
         POP      AF                  ; RESTORE CHARACTER
         CPL
         OUT      LDATA,A
         LD       A,2
         OUT      CMDP,A
         LD       A,0
         OUT      CMDP,A
       IF  NULLS EQ 0
         RET
       ENDIF
       IF  NULLS NE 0
         CP       LF                  ; CHECK FOR END OF LINE
         RET      NZ                  ; RETURN IF NOT LINE FEED CHAR.
         LD       A,NULLS+1           ; IF LF, GET NUMBER OF NULLS
LOUT50:  DEC      A                   ; CHECK FOR 0 NULLS AT TOP OF LOOP
         RET      Z                   ; RETURN IF ALL NULLS OUTPUT
         PUSH     AF                  ; SAVE NULLS COUNTER
         SUB      A                   ; PRINT A SINGLE NULL
         CALL     LOUT                ;   CHARACTER (RECURSIVE)
         POP      AF                  ; RESTORE NULLS COUNTER
         JR       LOUT50              ; LOOP TO PRINT NEXT NULL
       ENDIF
         EJECT
;
;
         TITLE    CLOCK ROUTINES
;
;
;   [DUMMY] TIME AND DATE ROUTINES
;
STRTCLK:
READCLK: RET                  ; TIME & DATE ROUTINES MAY BE ADDED HERE
;
         END
```

A.3

NLUN.Z80

ASSEMBLY LANGUAGE SUBROUTINES

```
        ENTRY TUART,STOP,ISTAT,ICR,MODSET,GETBUF,CLRBUF,SETTMB,WRTBUF
        ENTRY   $LUNTB,TTYOT1,TTYOT2,CURSAD,TERMIN
        EXT     $ERR,$DRV3,LPTDRV,DSKDRV,$BL,$BF,$IOERR
;
;       TABLE OF EQUATES
;
HOME:   EQU     0BH
LF:     EQU     0AH
FORM:   EQU     0CH
CR:     EQU     0DH
DEL:    EQU     7FH
IRECER: EQU     12H
FLKSTP: EQU     0
STAT1:  EQU     10H
STAT2:  EQU     20H
FKDATA: EQU     FLKSTP+1
DATA1:  EQU     STAT1+1
DATA2:  EQU     STAT2+1
CMDP:   EQU     24H
RDA:    EQU     40H
TBE:    EQU     80H
CLTX:   EQU     18H
CLTY:   EQU     19H
;
;       THIS TABLE DIRECTS THE PROGRAM TO
;       THE PROPER ROUTINE FOR THE LOGICAL UNIT
;       NUMBER SPECIFIED IN THE READ OR WRITE
;       INSTRUCTION
;
$LUNTB: DEFB    11              ;NUMBER OF ENTRIES + 1
;                               MAX 256
        DEFW    DRV1            ;LUN #1 PP1
        DEFW    DRV2            ;LUN #2 PP2
        DEFW    $DRV3           ;LUN #3 CONSOLE
        DEFW    FDRV            ;LUN #4 (FLUKE)
        DEFW    DSKDRV          ;LUN #5
        DEFW    DSKDRV          ;LUN #6
        DEFW    DSKDRV          ;LUN #7
        DEFW    DSKDRV          ;LUN #8 DISK FILES
        DEFW    DSKDRV          ;LUN #9
        DEFW    DSKDRV          ;LUN #10
;
; ***DISPATCH TABLE TERM PP1***
;
DRV1:   DEFW    DRV1FR  ;FORMATTED READ
        DEFW    DRV1FW  ;FORMATTED WRITE
        DEFW    DRV1BR  ;BINARY READ
        DEFW    DRV1BW  ;BINARY WRITE
        DEFW    DRV1RE  ;REWIND
        DEFW    DRV1BA  ;BACKSPACE
        DEFW    DRV1EN  ;ENDFILE
;
; ***DISPATCH TABLE TERM PP2***
;
DRV2:   DEFW    DRV1FR  ;FORMATTED READ
        DEFW    DRV2FW  ;FORMATTED WRITE
        DEFW    DRV1BR  ;BINARY READ
        DEFW    DRV1BW  ;BINARY WRITE
        DEFW    DRV1RE  ;REWIND
        DEFW    DRV1BA  ;BACKSPACE
        DEFW    DRV1EN  ;ENDFILE
;
```

```
;
;***REWIND, BACKSPACE, & ENDFILE***
;
DRV1RE:
DRV1BA:
DRV1EN: XOR     A               ;THESE OPERATIONS ARE
        RET                     ; NO-OPS FOR CRT
;
;
; ***BINARY READ & WRITE***
;
DRV1BR:
DRV1BW: JP      $IOERR          ;ILLEGAL OPERATIONS
                                ;   (PRINT ERROR AND RETURN)
;
; ***FORMATTED READ***
;
DRV1FR: JP      $IOERR          ;ILLEGAL OPERATIONS
                                ;   (PRINT ERROR AND RETURN)
;
; *** FORMATTED WRITE PP1 ***
;
DRV1FW: LD      A,($BL)                 ;GET BUFFER LENGTH
        OR      A                       ;BUFFER EMPTY?
        RET     Z                       ;IF SO DO NOTHING
        LD      HL,($BF)                ;POINT TO BUFFER
        DEC     A                       ;LENGTH -1
        PUSH    AF                      ;SAVE LENGTH
        LD      A,(HL)                  ;GET FIRST CHAR.
        CP      '+'                     ;+ MEANS NO CR/LF
        JP      Z,DR1FW2
        CP      '1'                     ;1 MEANS FORM FEED
        JP      NZ,DR1FW1
        LD      A,FORM                  ;
        CALL    TTYOT1                  ;SEND FORM FEED
        JP      DR1FW2                  ;
;
DR1FW1: LD      A,CR                    ;
        CALL    TTYOT1                  ;SEND CR/LF
        LD      A,LF                    ;
        CALL    TTYOT1                  ;
        LD      A,(HL)                  ;GET FIRST CHAR.
        CP      ' '                     ;SPACE MEANS 1 CR/LF
        JP      Z,DR1FW2
        CP      '0'                     ;0 MEANS 2 CR/LF
        JP      NZ,DR1FW2
        LD      A,CR                    ;
        CALL    TTYOT1                  ;SEND CR/LF
        LD      A,LF                    ;
        CALL    TTYOT1                  ;
;
DR1FW2: POP     AF                      ;GET LENGTH -1
        INC     HL                      ;POINT TO SECOND CHAR.
DR1FW3: RET     Z                       ;LAST CHAR. SENT
        PUSH    AF                      ;SAVE NO. OF CHAR. LEFT
        LD      A,(HL)                  ;GET NEXT CHAR.
        INC     HL                      ;BUMP POINTER
        CALL    TTYOT1                  ;SEND A CHAR.
        POP     AF                      ;GET COUNT
        DEC     A                       ;THIS SETS FLAGS
        JP      DR1FW3
```

```
;
; *** FORMATTED WRITE PP2 ***
;
DRV2FW: LD      A,($BL)
        OR      A
        RET     Z
        LD      HL,($BF)
        DEC     A
        PUSH    AF
        LD      A,(HL)
        CP      '+'
        JP      Z,DR2FW2
        CP      '1'
        JP      NZ,DR2FW1
        LD      A,FORM
        CALL    TTYOT2
        JP      DR2FW2
;
DR2FW1: LD      A,CR
        CALL    TTYOT2
        LD      A,LF
        CALL    TTYOT2
        LD      A,(HL)
        CP      ' '
        JP      Z,DR2FW2
        CP      '0'
        JP      NZ,DR2FW2
        LD      A,CR
        CALL    TTYOT2
        LD      A,LF
        CALL    TTYOT2
;
DR2FW2: POP     AF
        INC     HL
DR2FW3: RET     Z
        PUSH    AF
        LD      A,(HL)
        INC     HL
        CALL    TTYOT2
        POP     AF
        DEC     A
        JP      DR2FW3
;
;*** DEVICE DRIVER PP1 ***
;
TTYOT1: PUSH    BC
        PUSH    DE
        PUSH    HL
        PUSH    AF
        CALL    WRT1
        POP     AF
        POP     HL
        POP     DE
        POP     BC
        RET
;
RDY1:   IN      A,STAT1         ;PP1 TRANSMITTER BUFFER EMPTY?
        AND     TBE
        RET     Z               ;NO
        LD      A,-1
        RET                     ;YES
```

```
;
WRT1:    PUSH    AF
WRT10:   CALL    RDY1
         JR      Z,WRT10          ;LOOP UNTIL READY
         POP     AF
         OUT     DATA1,A          ;OUTPUT THE CHARACTER
         RET
;
;*** DEVICE DRIVER PP2 ***
;
TTYOT2:  PUSH    BC               ;SAME AS 1
         PUSH    DE
         PUSH    HL
         PUSH    AF
         CALL    WRT2
         POP     AF
         POP     HL
         POP     DE
         POP     BC
         RET
;
RDY2:    IN      A,STAT2
         AND     TBE
         RET     Z
         LD      A,-1
         RET
;
WRT2:    PUSH    AF
WRT20:   CALL    RDY2
         JR      Z,WRT20
         POP     AF
         OUT     DATA2,A
         RET
;
```

```
;
;       CURSOR ADDRESSING FUNCTION CALLED BY FORTRAN
;       IN THE FORM - CALL CURSAD(PP,Y,X)
;       X & Y ARE THE DECIMAL CO-ORDINATES WHERE THE
;       CURSOR SHOULD LAND ON THE SCREEN
;       X=0 TO 79
;       Y=0 TO 23
;
CURSAD: LD      A,(HL)          ;GET PLANT
        LD      (PLANT),A       ;SAVE IT
        EX      DE,HL
FILLBUF:LD      A,(HL)          ;GET Y VALUE
        LD      (BUF+3),A       ;PUT IT IN BUFFER
        PUSH    BC
        POP     HL
        LD      A,(HL)          ;GET X VALUE
        CALL    CONV            ;CONVERT IT
        LD      (BUF+1),A       ;PUT IT IN BUFFER
        LD      A,CLTX          ;PUT CURSOR ADDRESSING
        LD      (BUF),A         ;CHARACTERS IN BUFFER
        LD      A,CLTY
        LD      (BUF+2),A
        LD      HL,BUF          ;POINT TO FIRST CHAR.
        LD      B,5             ;4 CHAR.  +1
SEND:   DEC     B
        JR      Z,GOBAK         ;LAST CHAR?
        LD      A,(PLANT)       ;WHICH PLANT?
        CP      2
        JR      Z,SEND2
SEND1:  LD      A,-1            ;SET TERMINAL BUSY FLAG
        LD      (TMBSY1),A
        LD      A,(HL)          ;PP1
        CALL    TTYOT1
        JR      SKIP
SEND2:  LD      A,-1
        LD      (TMBSY2),A
        LD      A,(HL)          ;PP2
        CALL    TTYOT2
SKIP:   INC     HL              ;BUMP POINTER
        JR      SEND
GOBAK:  RET
;
PLANT:  DB      0
;
BUF:    DS      4
```

```
;
;PROGRAM TO CONVERT A BINARY NUMBER IN 'A' TO
;TWO DIGIT BCD NUMBER IN 'A'
;
CONV:   PUSH    BC
        LD      B,0             ;INIT TENS DIGIT COUNTER
LOOP:   SUB     10              ;INTEGER DEVIDE BY 10
        JR      C,OVER
        INC     B
        JR      LOOP
OVER:   ADD     10              ;RESTORE OVERFLOW
        LD      C,A             ;SAVE REMAINDER
        LD      A,B             ;GET TENS DIGIT
        ADD     A,A
        ADD     A,A             ;MULTIPLY IT BY 16
        ADD     A,A
        ADD     A,A
        ADD     A,C             ;ADD REMAINDER
        POP     BC
        RET
;
;
```

```
;
;  ***FLUKE DISPATCH TABLE***
;
FDRV:    DEFW    FDRVFR  ;FORMATTED READ
         DEFW    FDRVFW  ;FORMATTED WRITE
         DEFW    DRV1BR  ;BINARY READ
         DEFW    DRV1BW  ;BINARY WRITE
         DEFW    DRV1RE  ;REWIND
         DEFW    DRV1BA  ;BACKSPACE
         DEFW    DRV1EN  ;ENDFILE
;
;  ***FORMATTED READ***
;
FDRVFR:  XOR     A
         LD      ($BL),A          ;SET BUF. LENGTH=0
FDRFR1:  CALL    FLKIN            ;GET CHAR. FROM FLUKE
         AND     7FH              ;STRIP PARITY
         CP      LF               ;RET ON LF
         JR      Z,LOOPBK         ;END OF LINE
         PUSH    AF               ;SAVE THE CHAR.
         LD      HL,($BL)         ;GET CHAR. POS. IN BUF.
         LD      H,0              ;LIMIT TO 255
         EX      DE,HL            ;SAVE OFFSET IN DE
         LD      HL,($BF)         ;GET BUF. ADDRESS
         ADD     HL,DE            ;ADD OFFSET
         POP     AF               ;RESTORE CHAR.
         LD      (HL),A           ;PUT IT IN BUF.
         INC     DE               ;INC $BL
         EX      DE,HL
         LD      ($BL),HL         ;SAVE NEW VALUE
         LD      A,L
         CP      80H              ;CHECK FOR MAX LINE LENGTH
         JP      C,FDRFR1         ;GET NEXT CHAR
         CALL    $ERR             ;LINE TOO LONG
         DEFB    IRECER           ;ERROR CODE
         XOR     A                ;CLEAR FLAGS
         RET
LOOPBK:  LD      A,CR             ;TELL FORTRAN
         CP      CR               ;ITS END OF LINE
         RET
;
;  *** FORMATTED WRITE FLUKE  ***
;
FDRVFW:  LD      A,($BL)          ;GET BUFFER LENGTH
         OR      A                ;BUFFER EMPTY?
         RET     Z                ;IF SO DO NOTHING
         LD      HL,($BF)         ;POINT TO BUFFER
FDRFW3:  PUSH    AF               ;SAVE NO. OF CHAR. LEFT
         LD      A,(HL)           ;GET NEXT CHAR.
         INC     HL               ;BUMP POINTER
         CALL    FLKOUT           ;SEND A CHAR.
         POP     AF               ;GET COUNT
         DEC     A                ;THIS SETS FLAGS
         JR      NZ,FDRFW3        ;IF NOT LAST CHAR GO AROUND
         LD      A,CR             ;ELSE SEND CR AND RETURN
         CALL    FLKOUT
         RET
```

```
;
;*** FLUKE INPUT DRIVERS ***
;
FLKST:  IN      A,FLKSTP        ;FLUKE DATA AVAILABLE?
        AND     RDA
        RET     Z               ;NO
        LD      A,-1
        RET                     ;YES
;
;       ON ENTRY TO THIS ROUTINE
;       THE "CLEAR TO SEND" SIGNAL
;       IS SENT TO THE FLUKE
;       IT IS CLEARED BY THE START BIT
;       OF THE RECEIVED CHAR.
;
FLKIN:  LD      A,1
        OUT     CMDP,A          ;THIS PULSE SETS CTS
        LD      A,0
        OUT     CMDP,A
FKIN30: CALL    FLKST
        JR      Z,FKIN30        ;LOOP UNTIL CHAR. IS RECEIVED
        IN      A,FKDATA        ;GET IT
        AND     7FH             ;STRIP PARITY
        RET
;
;***FLUKE OUTPUT DRIVER***
;
FKRDY:  IN      A,FLKSTP        ;FLUKE TX BUFFER EMPTY?
        AND     TBE
        RET     Z               ;NO
        LD      A,-1
        RET                     ;YES
;
FLKOUT: PUSH    AF
FKOUT30:CALL    FKRDY
        JR      Z,FKOUT30       ;LOOP UNTIL READY
        POP     AF
        OUT     FKDATA,A        ;OUTPUT THE CHAR.
        RET
;
```

```
;
;                ***TUART***
;         CALLED BY FORTRAN IN FORM- CALL TUART
;         SETS UP INTERUPT VECTORS AND FLAGS
;
TUART:   DI                          ;INTS. OFF
         PUSH    AF                  ;SAVE REGS.
         PUSH    BC
         PUSH    DE
         PUSH    HL
         LD      HL,(6)              ;GET TOP OF MEM.
         LD      DE,-400H            ;-1K
         ADD     HL,DE               ;
         LD      A,H                 ;I REG. GETS PAGE
         LD      I,A
         LD      DE,10               ;INT. VECT=PAGE+10
         ADD     HL,DE
         LD      (STACK),SP          ;SAVE SP
         LD      SP,HL               ;USE STACK OPS
         LD      DE,ECHO1            ;TO PLACE ADDRESSES
         PUSH    DE                  ;IN VECTOR LOCATIONS
         LD      DE,16               ;SAVE FOR OTHER VECTOR
         ADD     HL,DE
         LD      SP,HL
         LD      DE,ECHO2
         PUSH    DE
         LD      SP,(STACK)          ;RESTORE SP.
         IM2                         ;SET INT MODE 2
         LD      A,10H               ;RDA INT FLAG
         OUT     13H,A               ;PP1
         OUT     23H,A               ;PP2
         POP     HL                  ;RESTORE REGS.
         POP     DE
         POP     BC
         POP     AF
         EI                          ;HOPE FOR THE BEST
         RET
```

```
;
;               ***STOP***
;       PROVIDES A SAFE EXIT FROM PROGRAM
;       USED BY ZAP INTERRUPT ROUTINE
;
STOP:   DI                          ;SAFE EXIT
        LD      A,0                 ;NO INT FLAGS
        OUT     13H,A               ;PP1
        OUT     23H,A               ;PP2
        JP      0                   ;BACK TO CDOS
```

```
;
;               ***ECHO 1 & 2***
;         ALL INTERRUPTS ENTER AT THESE LOCATIONS
;         ALL REGISTERS ARE SAVED
;
;         THE MODE IS SET BY THE FORTRAN PROGRAM
;         USING - CALL MODSET(PP,MODE)
;
;         THESE PROGRAMS WILL ACCEPT ONLY VALID
;         CHARACTERS FOR THE CURRENT MODE
;
;         IN MODE ZERO ONLY CONTROL CHARACTERS WILL
;         BE ACCEPTED (BELL RINGS FOR INVALID CHARACTERS)
;         THE STATUS FLAGS FOR THE CORRECT PLANT WILL
;         BE CHANGED ON RECEIPT OF A VALID CONTROL CHARACTER
;
;         IN MODE ONE ONLY NUMBERS,PERIOD,MINUS SIGN OR DELETE
;         WILL BE ACCEPTED.  VALID ENTRIES WILL BE DISPLAYED IN
;         THE UPPER RIGHT OF THE SCREEN AS SOON AS POSSIBLE.
;         DELETE WILL ERASE THE LAST CHARACTER ENTERED.
;
;         IN MODE TWO ONLY THE CHARACTER CONTROL Y WILL BE ACCEPTED
;         ALL OTHER CHARACTERS ARE INTERPRETED AS A NEGATIVE RESPONSE
;
;         IN MODE THREE ANY PRINTABLE CHARACTER OR DIGIT WILL BE
;         ACCEPTED
;
ECHO1:   PUSH     BC              ;ENTER HERE FROM TERM 1 INTERRUPT
         PUSH     DE
         PUSH     HL              ;SAVE REGS
         PUSH     AF
         LD       C,1             ;C GETS TERM ID
         IN       A,11H           ;A GETS INPUT CHAR
         AND      7FH             ;STRIP PARITY
         PUSH     AF              ;AND SAVE CHAR ON STACK
         LD       A,(MODE1)       ;GET INPUT MODE
         OR       A,A
         JP       Z,GETINT        ;MODE ZERO
         CP       1
         JP       Z,GETNUM        ;MODE ONE
         CP       2
         JP       Z,GETYN         ;MODE TWO
         CP       3
         JP       Z,GETANY        ;MODE THREE
         POP      AF
         JP       ECHOR           ;INVALID MODE
```

```
;
ECHO2:  PUSH    BC              ;ENTER HERE FROM TERM 2 INTERRUPT
        PUSH    DE
        PUSH    HL
        PUSH    AF
        LD      C,2             ;C GETS TERM ID
        IN      A,21H
        AND     7FH
        PUSH    AF
        LD      A,(MODE2)
        OR      A,A
        JP      Z,GETINT
        CP      1
        JP      Z,GETNUM
        CP      2
        JP      Z,GETYN
        CP      3
        JP      Z,GETANY
        POP     AF
ECHOR:  POP     AF              ;RECOVER REGS
        POP     HL
        POP     DE
        POP     BC
        EI                      ;RE-ENABLE INTERRUPTS
        RET
;
GETINT: POP     AF              ;ENTER HERE FOR MODE 0
        PUSH    AF              ;LEAVE COMMAND ON THE STACK
        CP      1               ;IS IT A VALID MODE 0 CHAR?
        JP      C,ERR           ;IF NOT RING THE BELL
        CP      1BH
        JP      NC,ERR
        LD      A,C             ;GET TERM ID
        CP      2
        JP      Z,STSET2
STSET1: POP     AF              ;GET COMMAND CODE
        LD      (STTUS1),A      ;STORE IT
        JP      ECHOR
STSET2: POP     AF
        LD      (STTUS2),A
        JP      ECHOR
ERR:    POP     AF              ;CLEAN UP STACK
        LD      A,C             ;GET TERM ID
        CP      2
        JP      Z,ERR2
ERR1:   LD      A,7             ;BELL
        LD      (BUF1+9),A      ;TO THE BUFFER
        CALL    WRTBAF          ;WRITE THE BUFFER
        JP      ECHOR
ERR2:   LD      A,7
        LD      (BUF2+9),A
        CALL    WRTBAF
        JP      ECHOR
```

```
;
GETNUM: POP     AF              ;ENTER HERE FOR MODE 1
        PUSH    AF
        CP      CR              ;IS IT A CARRIAGE RETURN
        JP      Z,GETN10
        CP      DEL             ;IS IT A DELETE CHAR.
        JP      Z,GETN20
        CP      '.'             ;IS IT A DECIMAL POINT
        JP      Z,GETN30
        CP      '-'             ;IS IT A MINUS SIGN
        JP      Z,GETN40
        CP      '0'             ;IS IT A VALID NUMBER
        JP      C,GETN50
        CP      ':'
        JP      NC,GETN50
        JP      GETN60
GETN10: POP     AF              ;ENTER HERE IF CR
        LD      A,C
        CP      2               ;WHICH PLANT
        JR      Z,GETN12
GETN11: LD      HL,-1
        LD      (CRFLG1),HL     ;SET PP1 CR FLAG
        JP      ECHOR
GETN12: LD      HL,-1
        LD      (CRFLG2),HL     ;SET PP2 CR FLAG
        JP      ECHOR
GETN20: POP     AF              ;ENTER HERE IF DELETE
        LD      A,C             ;WHICH PLANT
        CP      2
        JR      Z,GETN22
GETN21: CALL    POS1            ;IF FIRST CHARACTER
        CP      1               ;IN BUFFER DO NOTHING
        JP      Z,NODEC1
        DEC     HL              ;ELSE BACK UP POINTER
        LD      (POINT1),HL
NODEC1: LD      A,(HL)
        CP      '.'             ;IF DECIMAL ERASED
        JR      NZ,NODEC
        LD      A,0
        LD      (DOTFG1),A      ;CLEAR NO DOT FLAG
        JP      NODEC
GETN22: CALL    POS2            ;SAME AS ABOVE FOR PP2
        CP      1
        JP      Z,NODEC2
        DEC     HL
        LD      (POINT2),HL
NODEC2: LD      A,(HL)
        CP      '.'
        JR      NZ,NODEC
        LD      A,0
        LD      (DOTFG2),A
NODEC:  LD      A,' '           ;ERASE THE CHARACTER
        LD      (HL),A
        CALL    WRTBAF
        JP      ECHOR
```

```
GETN30: LD      A,C             ;ENTER HERE IF DECIMAL POINT
        CP      2
        JR      Z,GETN32
GETN31: LD      A,(DOTFG1)      ;ONLY ONE DECIMAL ALLOWED
        OR      A,A
        JP      NZ,GETN50       ;IF SECOND ONE RING BELL
        LD      A,-1
        LD      (DOTFG1),A      ;OK BUT NO MORE
        JP      GETN60
GETN32: LD      A,(DOTFG2)      ;SAME FOR PP2
        OR      A,A
        JP      NZ,GETN50
        LD      A,-1
        LD      (DOTFG2),A
        JP      GETN60
GETN40: LD      A,C             ;ENTER HERE IF MINUS SIGN
        CP      2
        JR      Z,GETN42
GETN41: CALL    POS1            ;ONLY ALLOWED IN FIRST POSITION
        CP      1
        JP      NZ,GETN50       ;IF NOT RING BELL
        JP      GETN60
GETN42: CALL    POS2            ;SAME FOR PP2
        CP      1
        JP      NZ,GETN50
        JP      GETN60
GETN50: POP     AF              ;ENTER HERE IF NOT A VALID CHAR.
        LD      A,C
        CP      2
        LD      A,7             ;GET A BELL READY
        JR      Z,GETN52
GETN51: LD      HL,(POINT1)
        JR      GETN55
GETN52: LD      HL,(POINT2)
GETN55: LD      (HL),A          ;RING IT
        CALL    WRTBAF
        JP      ECHOR
GETN60: LD      A,C             ;GOOD CHARACTER
        CP      2
        JR      Z,GETN62
GETN61: CALL    POS1            ;ANY ROOM IN BUFFER?
        CP      2
        JP      Z,GETN50        ;NO, RING BELL
        POP     AF
        LD      (HL),A          ;YES, DROP IT IN
        INC     HL
        LD      (POINT1),HL     ;AND BUMP POINTER
        CALL    WRTBAF
        JP      ECHOR
GETN62: CALL    POS2            ;SAME FOR PP2
        CP      2
        JP      Z,GETN50
        POP     AF
        LD      (HL),A
        INC     HL
        LD      (POINT2),HL
        CALL    WRTBAF
        JP      ECHOR
POINT1: DW      BUF1
POINT2: DW      BUF2
DOTFG1: DB      0
DOTFG2: DB      0
;
```

```
GETYN:  POP     AF              ;ENTER HERE IF MODE 2
        CP      CLTY            ;IS IT YES
        LD      HL,1            ;IF SO RETURN WITH POSITIVE
        JR      Z,YES           ;CARRIAGE RETURN FLAG
        LD      HL,-1           ;ELSE NEGATIVE FLAG
YES:    LD      A,C
        CP      2
        JR      Z,YES2
YES1:   LD      (CRFLG1),HL
        JP      ECHOR
YES2:   LD      (CRFLG2),HL
        JP      ECHOR
;
GETANY: POP     AF              ;ENTER HERE FOR MODE THREE
        PUSH    AF
        CP      CR
        JP      Z,GETN10
        CP      DEL
        JP      Z,GETN20        ;CR AND DEL SAME AS OTHER MODES
        CP      ' '             ;CHECK FOR PRINTABLE CHARACTERS
        JP      C,GETN50        ;IF NOT RING BELL
        CP      DEL
        JP      NC,GETN50
        JP      GETN60
;
;       CHECK FOR FIRST OR LAST POSITION
;       IN BUFFER
;       FIRST A=1,LAST A=2,ELSE A=0
;
;       RETURNS WITH HL AND DE = POINTER
;
POS1:   LD      A,0             ;PP1 ENTERS HERE
        LD      HL,BUF1         ;GET ADDRESS OF BUFFER 1
        LD      DE,(POINT1)     ;GET PRESENT POINTER 1
        OR      A               ;CLEAR CARRY
        SBC     HL,DE           ;CALCULATE POSITION
        JR      C,NFST1         ;NO CARRY MEANS FIRST POS.
        LD      A,1             ;IF FIRST A=1
NFST1:  LD      HL,BUF1+8       ;GET ADDRESS OF END OF BUF
        OR      A
        SBC     HL,DE           ;CALCULATE POSITION
        PUSH    DE
        POP     HL              ;HL=DE=POINTER 1
        RET     NC              ;NO CARRY MEANS NOT LAST EITHER
        LD      A,2             ;IF LAST A=2
        RET
POS2:   LD      A,0             ;SAME FOR PP2
        LD      HL,BUF2
        LD      DE,(POINT2)
        OR      A
        SBC     HL,DE
        JR      C,NFST2
        LD      A,1
NFST2:  LD      HL,BUF2+8
        OR      A
        SBC     HL,DE
        PUSH    DE
        POP     HL
        RET     NC
        LD      A,2
        RET
```

```
;
;       CALLED BY FORTRAN IN THE FORM- CALL MODSET(PP,MODE)
;       THIS SETS INPUT MODE AS EXPLAINED ABOVE
;
MODSET: LD      A,(HL)          ;GET PLANT
        CP      2
        JR      Z,MSET2
MSET1:  LD      A,(DE)          ;GET MODE
        LD      (MODE1),A       ;STORE IT
        OR      A,A
        RET     NZ              ;IS IT MODE 0
        LD      (STTUS1),A      ;IF SO STATUS GOES TO ZERO
        LD      (STTUS1+1),A
        RET
MSET2:  LD      A,(DE)          ;SAME FOR PP2
        LD      (MODE2),A
        OR      A,A
        RET     NZ
        LD      (STTUS2),A
        LD      (STTUS2+1),A
        RET
```

```
;
;       FORTRAN FUNCTION ISTAT(PP) RETURNS THE TYPE OF
;       INTERRUPT IN PROGRESS
;
ISTAT:  LD      A,(HL)          ;GET PLANT ID
        CP      2
        JR      Z,ISTAT2
ISTAT1: LD      HL,(STTUS1)     ;GET PLANT STATUS
        RET                     ;SEND IT BACK TO FORTRAN
ISTAT2: LD      HL,(STTUS2)
        RET
;
;       FORTRAN FUNCTION ICR(PP) RETURNS TRUE IF A CARRIAGE
;       RETURN CHARACTER HAS BEEN RECEIVED
;
ICR:    LD      A,(HL)
        CP      2
        JR      Z,ICR2
ICR1:   LD      HL,(CRFLG1)
        RET
ICR2:   LD      HL,(CRFLG2)
        RET
MODE1:  DB      0
MODE2:  DB      0
STTUS1: DW      0
STTUS2: DW      0
CRFLG1: DW      0
CRFLG2: DW      0
STACK:  DW      0
```

```
;
;       THIS IS THE ASSEMBLY LANGUAGE ENTRY TO THE
;       WRITE BUFFER ROUTINE
;
;       C CONTAINS THE CURRENT PLANT
;
;       IF THE TERMINAL IS NOT BUSY THE BUFFER WILL BE WRITTEN
;       IF NOT THE BUFFER READY FLAG IS SET SO THAT THE BUFFER
;       WILL BE WRITTEN AS SOON AS POSSIBLE
;
WRTBAF: LD      A,C             ;GET TERM ID
        CP      2
        JP      Z,WRTBA2
WRTBA1: LD      A,-1            ;SET BUFFER READY FLAG
        LD      (BRDY1),A
        LD      A,(TMBSY1)      ;TERMINAL BUSY?
        OR      A,A
        JP      Z,WRTB1         ;IF NOT WRITE THE BUFFER
        RET
WRTBA2: LD      A,-1
        LD      (BRDY2),A
        LD      A,(TMBSY2)
        OR      A,A
        JP      Z,WRTB2
        RET
```

```
;
;       THIS IS THE FORTRAN ENTRY POINT
;       THE TERMINAL BUSY FLAG IS RESET
;       AND THE BUFFER READY FLAG CHECKED
;       IF SET THE BUFFER IS WRITTEN
;
WRTBUF: LD      A,(HL)          ;ENTER HERE FROM FORTRAN
        CP      2
        JP      Z,WRTB2
WRTB1:  LD      A,-1            ;SET
        LD      (TMBSY1),A      ;TERMINAL BUSY FLAG
        LD      A,(BRDY1)       ;BUFFER READY?
        OR      A,A
        JR      Z,RDYL1
        LD      A,CLTY          ;POSITION CURSOR
        CALL    WRT1
        LD      A,0
        CALL    WRT1
        LD      A,CLTX
        CALL    WRT1
        LD      A,68H
        CALL    WRT1
        LD      HL,BUF1         ;IF SO WRITE IT
        LD      C,-10
OLOOP1: LD      A,(HL)
        CALL    WRT1
        CP      7               ;WAS IT A BELL CHAR?
        JR      NZ,NBEL1
        LD      A,' '           ;IF SO ERASE IT
        LD      (HL),A
NBEL1:  INC     HL
        INC     C
        JR      NZ,OLOOP1
        LD      A,0             ;CLEAR
        LD      (BRDY1),A       ;BUFFER READY FLAG
RDYL1:  CALL    RDY1
        JR      Z,RDYL1
        LD      A,0
        LD      (TMBSY1),A
        RET
WRTB2:  LD      A,-1            ;SAME AS PP1
        LD      (TMBSY2),A      ;SET TERMINAL BUSY FLAG
        LD      A,(BRDY2)
        OR      A,A
        JR      Z,RDYL2
        LD      A,CLTY          ;POSITION CURSOR
        CALL    WRT2
        LD      A,0
        CALL    WRT2
        LD      A,CLTX
        CALL    WRT2
        LD      A,68H
        CALL    WRT2
        LD      HL,BUF2
        LD      C,-10
```

```
OLOOP2: LD      A,(HL)
        CALL    WRT2
        CP      7
        JR      NZ,NBEL2
        LD      A,' '
        LD      (HL),A
NBEL2:  INC     HL
        INC     C
        JR      NZ,OLOOP2
        LD      A,0
        LD      (BRDY2),A
RDYL2:  CALL    RDY2
        JR      Z,RDYL2
        LD      A,0
        LD      (TMBSY2),A
        RET
```

```
;
;       THIS SUBROUTINE CAN BE CALLED BY FORTRAN
;       IN THE FORM - CALL SETTMB(PP)
;       IT WILL DISABLE THE WRTBAF ROUTINE UNTIL
;       A CALL WRTBUF(PP) IS EXECUTED
;
SETTMB: LD      A,(HL)          ;GET PLANT ID
        CP      2
        JP      Z,SETTM2
SETTM1: LD      A,-1            ;SET
        LD      (TMBSY1),A      ;TERMINAL BUSY FLAG
        RET
SETTM2: LD      A,-1
        LD      (TMBSY2),A
        RET
BUF1:   DB      'FIRST TIME'
BUF2:   DB      'FIRST TIME'
BRDY1:  DB      0
BRDY2:  DB      0
TMBSY1: DB      0
TMBSY2: DB      0
;
;       THIS IS CALLED BY FORTRAN IN THE FORM -CALL CLRBUF(PP)
;       IT CLEARS THE INPUT BUFFER FOR THE APPROPRIATE
;       PLANT AND ALSO WRITES OUT THE CLEAR BUFFER
;       ON THE SCREEN
;
CLRBUF: PUSH    HL
        LD      A,(HL)          ;GET PLANT ID
        CP      2
        JP      Z,CLR2
CLR1:   LD      A,0             ;CLEAR NO DOT FLAG
        LD      (DOTFG1),A
        LD      A,-1
        LD      (BRDY1),A       ;SET BUFFER READY FLAG
        LD      HL,BUF1         ;POINT TO BUFFER(PP)
        LD      (POINT1),HL
        JR      CLRB
CLR2:   LD      A,0             ;SAME AS PP1
        LD      (DOTFG2),A
        LD      A,-1
        LD      (BRDY2),A
        LD      HL,BUF2
        LD      (POINT2),HL
CLRB:   LD      C,-10           ;TEN CHARACTERS
        LD      A,' '           ;SET TO 'SPACE'
CLRLOP: LD      (HL),A
        INC     HL
        INC     C
        JR      NZ,CLRLOP
        POP     HL
        CALL    WRTBUF
        RET
```

```
;
;       CALLED BY FORTRAN IN THE FORM - CALL GETBUF(PP,LINE)
;       THIS SUBROUTINE RETURNS THE CURRENT CONTENTS
;       OF THE INPUT BUFFER TO THE CHARACTER ARRAY
;       POINTED TO BY THE SECOND PARAMETER
;
GETBUF: LD      A,(HL)          ;GET PLANT ID
        CP      2
        JP      Z,GETB2
GETB1:  LD      HL,BUF1         ;DE POINTS TO FORTRAN LOCATION
        LD      BC,10           ;10 CHARACTERS
        LDIR                    ;BLOCK MOVE
        LD      HL,0
        LD      (CRFLG1),HL     ;RESET CR FLAG
        RET
GETB2:  LD      HL,BUF2         ;SAME AS PP1
        LD      BC,10
        LDIR
        LD      HL,0
        LD      (CRFLG2),HL
        RET
;
;       CALLED BY FORTRAN IN THE FORM - CALL TERMIN
;       THIS SUBROUTINE ASSURES THAT ALL THE SERIAL
;       INTERFACES ARE SET TO THE CORRECT BAUD RATE
;
TERMIN: PUSH AF
        LD      A,9             ;RESET ALL THE UARTS
        OUT     2,A
        OUT     12H,A
        OUT     22H,A
        LD      A,88H           ;SET FLUKE TO 1200 BAUD
        OUT     0,A
        LD      A,0C0H          ;AND THE REST TO 9600
        OUT     10H,A
        OUT     20H,A
        POP     AF
        RET
;
        END
```

A.4

COMMON.RFR

```
#
      LOGICAL TIME,MSG
      INTEGER PLANT,PAGE,X,Y,AVEN,SECT,WORD,PRIOR,LONG,SHORT,FEDLOC,HTCLOC
      INTEGER DIFLOC,GASLOC,HR,MIN,SEC,PG,COUNT,WTSREM,OLDSW,DAY,YEAR,OLDHR
      INTEGER STAT,TMSTRT,NRUN
      REAL A,B,CONST,VAL,CENTER,SPAN,INITWT,OLDWT,FEDTOT,TOT,TMSEC,TARE,SAMPL
      REAL ACCWT,BARWT
      DOUBLE PRECISION DESC,RUN
#
#
      COMMON /CHAN/PLANT(224),PAGE(224),X(224),Y(224),AVEN(224),SECT(224),
     1           WORD(224),PRIOR(224),LONG(500),SHORT(100),FEDLOC(3,2),
     2           HTCLOC(32),DIFLOC(24),GASLOC(4,2),A(224),B(224),CONST(10),
     3           DESC(224),TIME(12),MSG(25),HR,MIN,SEC,PG(2),COUNT(2)
#
      COMMON/FLAGS/VAL(224),CENTER(224),SPAN(224),INITWT(2,2),WTSREM(2),OLDS
     1           OLDWT(2),FEDTOT(2),TOT(2,5),TMSEC(2,5),TARE(2,2),SAMPLE(2,
     2           ACCWT(2,2),BARWT(2,2),DAY,YEAR,OLDHR,STAT(2),RUN(2),NRUN(2
#
```

A.5

COMBLOK.RFR

```
B.
#
      LOGICAL TIME,MSG
      INTEGER HR,MIN,SEC,PG,COUNT,FBLOK
      REAL BLOCK
#
      COMMON/CHAN/BLOCK(2137),TIME(12),MSG(25),HR,MIN,SEC,PG(2),COUNT(2)
#
      COMMON/FLAGS/FBLOK(1451)
#
```

A.6

DISP.RFR

```
        PROGRAM MAIN
      DEFINE( FLUKE, 4)
      DEFINE( SPACE, 32)
      DEFINE( CHRA, 65)
      DEFINE( ENDOFSCAN, 20)
      DEFINE( ESC, 27)
#
      INTEGER PP, I, CHN, N, LONGC, SHORTC
#
      INCLUDE COMMON.RFR
#
#     THIS PART OF THE PROGRAM IS ONLY EXECUTED
      ONCE EACH TIME POWER IS APPLIED TO THE SYSTEM
#
#     INITIALIZE ALL UARTS
#
      CALL TERMIN
#
#     CLEAR BOTH SCREENS AND BUFFERS
#
      FOR (PP=1;PP<=2;PP=PP+1)
          [
          CALL CLEAR(PP)
          CALL CLRBUF(PP)
          COUNT(PP)=0
          ]
#     INITIALIZE ARRAYS
      CALL GETBLK
#     INITIALIZE FLUKE
      CALL FINIT
#     WRITE PAGE 0 TABLE TO
#     EACH PLANT
      FOR (PP=1;PP<=2;PP=PP+1)
          [
          PG(PP)=0
          CALL FILL(PP)
          CALL MODSET(PP,0)
          ]
#     SET UP INTERRUPTS
      CALL TUART
#     INITIALIZE LIST COUNTS
      LONGC=1
      SHORTC=1
#
```

```
#
#    MAIN LOOP
#    THIS IS AN INFINITE LOOP AND WILL CONTINUE
#    UNTIL IT IS STOPPED BY THE 'ZAP' INTERRUPT
#    OR THE POWER IS SHUT OFF
#
     REPEAT
          [
          CHN=SHORT(SHORTC)
          CALL DOCHN(CHN)
          SHORTC=SHORTC+1
          IF(SHORT(SHORTC)>223)
               SHORTC=1
          CHN=LONG(LONGC)
          CALL DOCHN(CHN)
          LONGC=LONGC+1
          IF(LONG(LONGC)>223)
               LONGC=1
          IF(HR^=OLDHR)
               [
               IF(HR<OLDHR)
                    DAY=DAY+1
               FOR(PP=1;PP<=2;PP=PP+1)
                    [
                    IF(STAT(PP)^=0)
                         CALL HRLY(PP,0)
                    ]
               OLDHR=HR
               IF(STAT(1)^=0\STAT(2)^=0)
                    CALL DUMPF
               ]
          ]
     END
#
```

A.7

SUBS.RFR

```
#
      SUBROUTINE DOCHN(CHN)
#
    INTEGER CHN,PP
#
#   THIS SUBROUTINE PERFORMS THE NECESSARY
#   PROCESSING FOR EACH CHANNEL OR CALCULATION
#   AND THEN SERVICES THE INTERRUPTS BEFORE RETURNING
#   TO THE MAIN LOOP FOR THE NEXT CHANNEL
#
    IF(CHN>179)
        CALL ONLINE(CHN)
    ELSE
        CALL AVCHN(CHN)
    CALL DISP(CHN)
    FOR (PP=1;PP<=2;PP=PP+1)
        [
        IF (ISTAT(PP)^=0)
            CALL DISPAT(PP)
        ]
    RETURN
    END
```

```
#
          SUBROUTINE CLEAR(PP)
#
#    THIS SUBROUTINE CLEARS THE SCREEN
#    AND HOMES THE CURSOR
#    A DELAY CHARACTER IS REQUIRED
#    AT 9600 BAUD FOR THIS FUNCTION
#
     LOGICAL CLR,DLY
     INTEGER PP
     DLY=27
     CLR=12
     WRITE (PP,10)CLR,DLY,DLY
10   FORMAT ('+',3A1)
     RETURN
     END
#
          SUBROUTINE DUMPF
#
     INCLUDE COMBLOK.RFR
#
#    THIS SUBROUTINE STORES ALL THE FLAGS WHICH
#    ARE REQUIRED TO RECOVER FROM A POWER FAILURE
#
     ENCODE(MSG,10)
10   FORMAT('DUMPING FLAGS')
     CALL PROGST
     CALL OPEN(7,'DISP    DAT',0)
     FOR (I=0;I<=21;I=I+1)
          [
          J=I+68
          K=I*64+1
          L=K+63
          WRITE(7,REC=J)(FBLOK(M),M=K,L)
          ]
     WRITE(7,REC=90)(FBLOK(M),M=1409,1451)
     ENDFILE 7
     RETURN
     END
#
```

```
#
      SUBROUTINE GETBLK
#
    INCLUDE COMBLOK.RFR
#
#   THIS SUBROUTINE READS THE BLOCK OF DATA
#   CONTAINED IN FILE "DISP.DAT" WHICH MUST
#   RESIDE ON DRIVE A:
#
    CALL OPEN(7,'DISP    DAT',0)
    FOR (I=0;I<=65;I=I+1)
        [
        J=I+1
        K=I*32+1
        L=K+31
        READ(7,REC=J)(BLOCK(M),M=K,L)
        ]
    READ(7,REC=67)(BLOCK(M),M=2113,2137)
    FOR (I=0;I<=21;I=I+1)
        [
        J=I+68
        K=I*64+1
        L=K+63
        READ(7,REC=J)(FBLOK(M),M=K,L)
        ]
    READ(7,REC=90)(FBLOK(M),M=1409,1451)
    ENDFILE 7
    RETURN
    END
#
```

```
#
      SUBROUTINE FILL(PP)
#
    LOGICAL HON,HOFF,BON,BOFF,TEMP(8),VAL$(6),CENT$(6)
    INTEGER PP,X1,Y1,I,N,K
    REAL SP
#
    INCLUDE COMMON.RFR
#
#   THIS SUBROUTINE WRITES THE INFORMATION
#   FOR THE CURRENTLY SELECTED PAGE TO
#   THE SCREEN.  ALARM CONDITIONS ARE INDICATED
#   BY A BLINKING NUMBER
#
    CALL CLEAR(PP)
    ENCODE(MSG,5)PP
5   FORMAT('FILLING PLANT',I2,10X)
    ENCODE(TIME,7)
7   FORMAT('NOT AVAILABL')
    CALL PROGST
    X1=0
    Y1=6
    CALL CURSAD(PP,Y1,X1)
    WRITE(PP,8)PG(PP)
8   FORMAT('+',35('-'),'PAGE ',I3,37('-'))
    HON=2
    HOFF=4
    FOR (I=0; I<224; I=I+1)
        [
        N=I+1
        IF (PLANT(N)==PP&PAGE(N)==PG(PP))
            [
            SP=ABS(SPAN(N))
            IF (VAL(N)<(CENTER(N)-SP)\VAL(N)>(CENTER(N)+SP))
                [
                BON=5
                BOFF=1
                ]
            ELSE
                [
                BON=27
                BOFF=27
                ]
            X1=(X(N)-1)*27
            Y1=Y(N)+6
            CALL LEFTJ(CENT$,CENTER(N))
            CALL LEFTJ(VAL$,VAL(N))
            CALL CURSAD(PP,Y1,X1)
            ENCODE (TEMP,10)DESC(N)
10          FORMAT (A8)
            WRITE (PP,20)HON,I,(TEMP(K),K=1,6),HOFF,BON,VAL$,BOFF,HON,CENT
                    (TEMP(K),K=7,8),HOFF
20          FORMAT ('+',A1,I3,1X,22A1,1X,3A1)
            ]
        ]
    RETURN
    END
#
```

```
#
      SUBROUTINE DISP(CHN)
#
     LOGICAL BELL, BON, BOFF, VAL$(6)
     INTEGER PP, X1, Y1, CHN, N
     REAL SP
#
     INCLUDE COMMON.RFR
#
#    THIS SUBROUTINE WRITES THE CURRENT VALUE
#    OF A CHANNEL OR CALCULATION RESULT TO
#    THE APPROPRIATE POSITION IN THE DISPLAY
#    TABLE.  A BLINKING NUMBER ACCOMPANIED BY A
#    BELL CHARACTER INDICATES AN ALARM CONDITION.
#
     N=CHN+1
     PP=PLANT(N)
     IF (PAGE(N)==PG(PP))
          [
          SP=ABS(SPAN(N))
          IF (VAL(N)<(CENTER(N)-SP)\VAL(N)>(CENTER(N)+SP))
               [
               BELL=7
               BON=5
               BOFF=1
               ]
          ELSE
               [
               BELL=27
               BON=27
               BOFF=27
               ]
          X1=(X(N)-1)*27+10
          Y1=Y(N)+6
          CALL LEFTJ(VAL$, VAL(N))
          CALL CURSAD(PP, Y1, X1)
          WRITE (PP, 20)BELL, BON, VAL$, BOFF
20        FORMAT ('+', 9A1)
          CALL WRTBUF(PP)
          ]
     RETURN
     END
#
```

```
#
          SUBROUTINE PROGST
#
      INTEGER PP, I, X1, Y1
#
      INCLUDE COMMON.RFR
#
#     THIS SUBROUTINE WRITES THE CURRENT PLANT AND
#     PROGRAM STATUS TO THE DISPLAY SCREENS
#
      FOR (PP=1; PP<=2; PP=PP+1)
          [
          IF(PG(PP)<4)
              [
              X1=0
              Y1=2
              CALL CURSAD(PP, Y1, X1)
              WRITE (PP, 30)TIME
30            FORMAT ('+TIME:  ', 12A1)
              CALL WRTBUF(PP)
              Y1=3
              CALL CURSAD(PP, Y1, X1)
              WRITE (PP, 10)
10            FORMAT ('+PROGRAM STATUS:')
              CALL WRTBUF(PP)
              Y1=4
              CALL CURSAD(PP, Y1, X1)
              WRITE (PP, 20)MSG
20            FORMAT ('+', 25A1)
              CALL WRTBUF(PP)
              X1=27
              Y1=2
              CALL CURSAD(PP, Y1, X1)
              WRITE(PP, 40)YEAR
40            FORMAT('+YEAR:  ', I5)
              CALL WRTBUF(PP)
              Y1=3
              CALL CURSAD(PP, Y1, X1)
              WRITE(PP, 50)
50            FORMAT('+PLANT STATUS:')
              CALL WRTBUF(PP)
              Y1=4
              CALL CURSAD(PP, Y1, X1)
              WRITE(PP, 60)NRUN(PP), RUN(PP)
60            FORMAT('+RUN:', I4, 2X, A8)
              CALL WRTBUF(PP)
              ]
          ]
      FOR (I=1; I<=25; I=I+1)
          MSG(I)=SPACE
      RETURN
      END
#
```

```
#
      SUBROUTINE LEFTJ(STRNG,NUM)
#
    LOGICAL STRNG(6)
    REAL NUM,ANUM
#
#   THIS SUBROUTINE RECEIVES A REAL NUMBER FROM
#   THE CALLING PROGRAM AND RETURNS A CHARACTER
#   ARRAY IN THE MOST APPROPRIATE FORMAT FOR DISPLAY
#   IN THE TABLE.
#
    ANUM=ABS(NUM)
    IF (ANUM>32000. )
        ENCODE (STRNG,70)
    ELSE IF (INT(ANUM)==7777)
        ENCODE (STRNG,60)
    ELSE IF (NUM==0)
        ENCODE (STRNG,50)
    ELSE IF (ANUM<.99995)
        ENCODE (STRNG,5)NUM
    ELSE IF (ANUM<9.9995)
        ENCODE (STRNG,10)NUM
    ELSE IF (ANUM<99.995)
        ENCODE (STRNG,20)NUM
    ELSE IF (ANUM<999.95)
        ENCODE (STRNG,30)NUM
    ELSE IF (ANUM<=32000)
        [
        I=NUM
        ENCODE (STRNG,40)I
        ]
5   FORMAT (F6.4)
10  FORMAT (F6.3)
20  FORMAT (F6.2)
30  FORMAT (F6.1)
40  FORMAT (I6)
50  FORMAT ('    0.0')
60  FORMAT ('     BT')
70  FORMAT ('TOO BG')
    RETURN
    END
```

A.8

STORE.RFR

```
      SUBROUTINE HRLY(PP,MODE)
#
      LOGICAL NAME(11)
      INTEGER PP,WORD1,SECT1,CHN,N,DRIVE,I,J,MODE
      INTEGER FSTAT,FDAY,FHR,FRUN
      REAL STORAR
#
      INCLUDE COMMON.RFR
#
      COMMON/STOR/STORAR(32,4)
#
      COMMON/DIR/FSTAT(128),FDAY(128),FHR(128),FRUN(128)
#
#     THIS SUBROUTINE WRITES AN HOURLY RECORD TO THE APPROPRIATE
#     DISK.  IT IS ALSO USED TO WRITE A STARTUP OR SHUTDOWN RECORD
#
      FOR(I=1;I<=4;I=I+1)
          FOR(J=1;J<=32;J=J+1)
              STORAR(J,I)=0.0
      FOR(CHN=0;CHN<=223;CHN=CHN+1)
          [
          N=CHN+1
          IF(PLANT(N)==PP)
              [
              SECT1=SECT(N)
              WORD1=WORD(N)
              STORAR(WORD1,SECT1)=VAL(N)
              ]
          ]
      DRIVE=PP+1
      ENCODE(NAME,100)PP
100   FORMAT('PLANT',I1,'  DAT')
      CALL OPEN(7,NAME,DRIVE)
      CALL GDIR
      FOR(N=1;FSTAT(N)^=0&N<=128;N=N+1)
          ;
      IF(N==129)
          [
          CALL CLRTOP(PP)
          WRITE(PP,110)
110       FORMAT('+DIRECTORY ERROR')
          CALL WRTBUF(PP)
          ]
      ELSE
          [
          CALL STORE(PP,N)
          FSTAT(N)=1
          FDAY(N)=DAY
          IF(MODE==0)
              FHR(N)=HR
          ELSE
              FHR(N)=MODE
          FRUN(N)=NRUN(PP)
          CALL WDIR
          ]
      ENDFILE 7
      RETURN
      END
#
```

```
#
      SUBROUTINE STORE(PP,N)
#
      INTEGER N,PP,K,J
      REAL STORAR,REST
#
      INCLUDE COMMON.RFR
#
      COMMON/STOR/STORAR(32,3),REST(16)
#
#     THIS SUBROUTINE DOES THE ACTUAL WRITING OF THE
#     HOURLY RECORD.  THE SECTOR WHERE THE MANUALLY ENTERED
#     DATA WILL BE WRITTEN IS CLEARED AT THE SAME TIME SO
#     THAT ALL THE SECTORS WILL WILL CONTAIN SOME DATA
#
      FOR(I=1;I<=3;I=I+1)
          [
          K=(N-1)*5+8+I
          WRITE(7,REC=K)(STORAR(J,I),J=1,32)
          ]
      K=(N-1)*5+8+4
      WRITE(7,REC=K)REST,(INITWT(PP,J),J=1,2),WTSREM(PP),OLDSW(PP),OLDWT(PP
          FEDTOT(PP),(TARE(PP,J),J=1,2),(SAMPLE(PP,J),J=1,2),(ACCWT(PP,J),J
          (BARWT(PP,J),J=1,2),YEAR,DAY,STAT,TMSEC(PP,5)
      FOR(I=1;I<=32;I=I+1)
          STORAR(I,1)=0.
      K=(N-1)*5+8+5
      WRITE(7,REC=K)(STORAR(J,1),J=1,32)
      RETURN
      END
```

A.9

FLUKE.RFR

```
#
      SUBROUTINE FINIT
#
    LOGICAL LINE(30),CMDSTR(20)
    INTEGER I,J,K,D,H,M,S
#
    INCLUDE COMMON.RFR
#
#   THIS SUBROUTINE WILL RESET THE FLUKE
#   TO A KNOWN STATE
#
    ENCODE (CMDSTR,10)
10  FORMAT ('E1E1W4X0')
    FOR (I=1;I<=7;I=I+2)
        [
        J=I+1
        ENCODE (LINE,20) (CMDSTR(K),K=I,J)
20          FORMAT (2A1,' ')
        CALL COMAND(LINE)
        ]
    ENCODE (CMDSTR,30)
30  FORMAT ('M001F001L179')
    FOR (I=1;I<=9;I=I+4)
        [
        J=I+3
        ENCODE (LINE,40) (CMDSTR(K),K=I,J)
40      FORMAT (4A1,1X)
        CALL COMAND(LINE)
        ]
#
#       START AND STOP SCAN
#       TO SET FLUKE IN KNOWN STATE
#
    ENCODE (LINE,50)
50  FORMAT ('S1',1X)
    CALL COMAND(LINE)
    REPEAT
        [
        READ(FLUKE,60)LINE
60      FORMAT(30A1)
        CALL ZAPLF(LINE)
        IF (LINE(1)==CHRY)
            [
            FOR (I=1;I<=12;I=I+1)
                TIME(I)=LINE(I+1)
            ]
        ELSE IF (LINE(1)==CHRD)
            [
            DECODE (LINE,65)HR,MIN,SEC
65          FORMAT (5X,I2,1X,I2,1X,I2)
            ]
        ] UNTIL (LINE(1)==CHRA)
    ENCODE (LINE,70)
70  FORMAT ('S0 ')
    CALL COMAND(LINE)
    CALL NDSCAN
    RETURN
    END
#
```

```
#
        SUBROUTINE SETCLK
#
      LOGICAL LINE(30)
      INTEGER I,J,K,D,H,M,S
#
#
      INCLUDE COMMON.RFR
#
#     THIS SUBROUTINE SETS THE CLOCK ON THE FLUKE
#
      S=SEC+100
      M=MIN+100
      H=HR+100
      D=DAY+1000
      ENCODE(LINE,80)S
80    FORMAT(10X,I3,1X)
      ENCODE(LINE,82)M
82    FORMAT(7X,I3,',')
      ENCODE(LINE,84)H
84    FORMAT(4X,I3,',')
      ENCODE(LINE,86)D
86    FORMAT(I4,',')
      ENCODE(LINE,88)
88    FORMAT('T')
      CALL COMAND(LINE)
      RETURN
      END
#
```

```
#
      SUBROUTINE COMAND(LINE)
#
    LOGICAL LINE(30)
    INTEGER I,K
#
#   THIS SUBROUTINE WILL SEND A COMMAND TO
#   THE FLUKE FOLLOWED BY A CARRIAGE RETURN BUT
#   NO LINE FEED.  THE FLUKE'S RESPONSE IS RETURNED
#   IN ARRAY LINE
#
    FOR (K=1;LINE(K)^=SPACE & K<=30;K=K+1)
        ;
    IF (K^=1)
        K=K-1
    WRITE (FLUKE,10) (LINE(I),I=1,K)
10  FORMAT (30A1)
    READ(FLUKE,20)LINE
20  FORMAT(30A1)
    CALL ZAPLF(LINE)
    RETURN
    END
#
```

```
#
      SUBROUTINE NDSCAN
#
      LOGICAL LINE(30)
#
#     THIS SUBROUTINE READS THE RESPONSE FROM
#     THE FLUKE UNTIL AN END OF SCAN CHARACTER
#     IS RECEIVED.
#
      REPEAT
          [
          READ (FLUKE,10) LINE
10        FORMAT (30A1)
          CALL ZAPLF(LINE)
          ] UNTIL (LINE(1)==ENDOFSCAN)
      RETURN
      END
#
```

```
#
      SUBROUTINE AVCHN(CHN)
#
      LOGICAL LINE(30)
      INTEGER CHN,REP,J,PP,I,N
      REAL TEMP
#
      INCLUDE COMMON.RFR
#
#     THIS SUBROUTINE PUTS THE FLUKE INTO A MONITOR
#     SCAN OF THE APPROPRIATE CHANNEL.  IT AVERAGES
#     THE READINGS OVER THE REQUIRED NUMBER OF TIMES
#     THEN MULTIPLIES BY THE VALUE OF 'A' AND ADDS THE
#     VALUE OF 'B'
#
      ENCODE(MSG,10)CHN
10    FORMAT('READING CHANNEL',I4)
      CALL PROGST
      J=CHN+1000
      ENCODE (LINE,20)J
20    FORMAT (I4,16X)
      ENCODE (LINE,30)
30    FORMAT ('M')
      CALL COMAND(LINE)
      ENCODE (LINE,40)
40    FORMAT ('S3 ')
      CALL COMAND(LINE)
      VAL(CHN+1)=0.0
      REP=AVEN(CHN+1)
      REPEAT
          [
          READ (FLUKE,50)LINE
50        FORMAT (30A1)
          CALL ZAPLF(LINE)
          IF (LINE(1)==CHRY)
              [
              FOR (I=1; I<=12; I=I+1)
                  TIME(I)=LINE(I+1)
              ]
          ELSE IF (LINE(1)==CHRD)
              [
              DECODE (LINE,70)HR,MIN,SEC
70            FORMAT (5X,I2,1X,I2,1X,I2)
              ]
          ELSE IF (LINE(1)==CHRA)
              [
              DECODE (LINE,80)TEMP
80            FORMAT (5X,F7.2)
              VAL(CHN+1)=VAL(CHN+1)+TEMP
              REP=REP-1
              ]
          ] UNTIL (REP==0)
      VAL(CHN+1)=VAL(CHN+1)/AVEN(CHN+1)
      VAL(CHN+1)=VAL(CHN+1)*A(CHN+1)+B(CHN+1)
      ENCODE (LINE,90)
90    FORMAT ('S0 ')
      CALL COMAND(LINE)
      CALL NDSCAN
      RETURN
      END
#
```

```
#
      SUBROUTINE ZAPLF(LINE)
#
#     THIS SUBROUTINE REMOVES THE LINE FEED CHARACTER
#     WHICH THE FLUKE SENDS AFTER ITS RESPONSE
#
      LOGICAL LINE(30)
      IF (LINE(1)==CHRLF)
          [
          FOR(I=1; I<30; I=I+1)
              LINE(I)=LINE(I+1)
          ]
      RETURN
      END
```

A.10

DISPAT.RFR

```
#
      SUBROUTINE DISPAT(PP)
#
    INTEGER N, ISTAT, PP, MODE
#
    INCLUDE COMMON.RFR
#
#   THIS SUBROUTINE IS CALLED AFTER EACH ITEM IN THE
#   LONG OR SHORT LISTS IS PROCESSED AND THERE IS AN
#   INTERRUPT IN PROGRESS.
#   IF THE INTERRUPT ROUTINE HAS JUST BEEN REQUESTED
#   (IE. COUNT(PP)=0 ) OR A CARRIAGE RETURN HAS BEEN
#   RECEIVED THIS ROUTINE DISPATCHES THE PROGRAM TO
#   THE APPROPRIATE ROUTINE. OTHERWISE THIS ROUTINE
#   SIMPLY RETURNS CONTROL TO THE CALLING PROGRAM
#
    IF(COUNT(PP)==0\ICR(PP)^=0)
        [
        N=ISTAT(PP)
        GO TO (10, 20, 30, 40, 50, 60, 70, 80, 90), N
        N=N-9
        GO TO (100, 110, 120, 130, 140, 150, 160, 170, 180), N
        N=N-9
        GO TO (190, 200, 210, 220, 230, 240, 250, 260), N
        GO TO 270
        ]
    RETURN
```

```
 10  CALL AUTOI(PP)
     RETURN
 20  CALL BARREL(PP)
     RETURN
 30  CALL DUMMY(PP)
     RETURN
 40  CALL DAYIN(PP)
     RETURN
 50  CALL EDIT(PP,1)
     RETURN
 60  CALL DUMMY(PP)
     RETURN
 70  CALL DUMMY(PP)
     RETURN
 80  CALL DUMMY(PP)
     RETURN
 90  CALL DUMMY(PP)
     RETURN
100  CALL DUMMY(PP)
     RETURN
110  CALL DUMMY(PP)
     RETURN
120  CALL DUMMY(PP)
     RETURN
130  CALL DUMMY(PP)
     RETURN
140  CALL DUMMY(PP)
     RETURN
150  CALL OTHER(PP)
     RETURN
160  CALL DUMMY(PP)
     RETURN
170  CONTINUE
     IF(STAT(PP)^=0)
         CALL SHTDWN(PP)
     ELSE
         CALL DUMMY(PP)
     RETURN
180  CONTINUE
     IF(STAT(PP)==0)
         CALL STRTUP(PP)
     ELSE
         CALL DUMMY(PP)
     RETURN
190  CALL DUMMY(PP)
     RETURN
200  CALL EDIT(PP,2)
     RETURN
210  CALL UNCOV(PP)
     RETURN
220  CALL DUMMY(PP)
     RETURN
230  CALL WRTMAN(PP)
     RETURN
240  CALL XCHNGD(PP)
     RETURN
250  CALL DUMMY(PP)
     RETURN
260  CALL ZAP(PP)
     RETURN
270  RETURN
     END
#
```

```
#
      SUBROUTINE DUMMY(PP)
#
    LOGICAL BELL
    INTEGER PP
#
    INCLUDE COMMON.RFR
#
#   IF AN UNIMPLEMENTED OR INVALID INTERRUPT COMMAND
#   IS REQUESTED THIS SUBROUTINE SENDS A BELL CHARACTER
#   TO THE OFFENDING TERMINAL AND RESETS THE INTERRUPT
#   FLAGS
#
    BELL=7
    CALL CLRTOP(PP)
    WRITE(PP,10)BELL
10  FORMAT('+',A1)
    CALL WRTBUF(PP)
    CALL MODSET(PP,0)
    COUNT(PP)=0
    RETURN
    END
#
```

```
#
      SUBROUTINE OTHER(PP)
#
      INTEGER PP
#
      INCLUDE COMMON.RFR
#
#     THIS INTERRUPT SUBROUTINE INCREMENTS THE
#     PAGE NUMBER OF THE APPROPRIATE PLANT
#     AND WRITES THE NEW PAGE TO THE SCREEN
#
      PG(PP)=PG(PP)+1
      IF(PG(PP)>2)
          PG(PP)=0
      CALL FILL(PP,PG(PP))
      CALL MODSET(PP,0)
      CALL WRTBUF(PP)
      RETURN
      END
#
```

```
#
        SUBROUTINE AUTOI(PP)
#
    INTEGER PP, I
#
    INCLUDE COMMON.RFR
#
#   THIS INTERRUPT SUBROUTINE CHECKS EACH OF THE 224 CHANNELS
#   AND CALCULATION NUMBERS.  IF THE CHANNEL IS ASSOCIATED WITH
#   THE PLANT IN QUESTION AND IT HAS A NEGATIVE TOLERANCE VALUE
#   THE CENTER POINT FOR THAT CHANNEL WILL BE SET EQUAL TO THE
#   CURRENT VALUE.  THE NEW TABLE IS THEN WRITTEN TO THE SCREEN.
#
    FOR (I=1; I<=224; I=I+1)
        [
        IF (SPAN(I)<0.0&PP==PLANT(I))
            CENTER(I)=VAL(I)
        ]
    CALL FILL(PP,PG(PP))
    CALL MODSET(PP,0)
    CALL WRTBUF(PP)
    CALL DUMPF
    RETURN
    END
#
```

```
#
        SUBROUTINE BARREL(PP)
#
    LOGICAL LINE(10)
    INTEGER PP,BAREL(2),BAR,BAR$(2),J,X,Y
    REAL TEMP,FINWT(2)
#
    INCLUDE COMMON.RFR
#
#   THIS INTERRUPT SUBROUTINE COLLECTS THE INFORMATION
#   REQUIRED BY THE PROGRAM WHEN A LE OR HE BARREL IS
#   CHANGED.
#
    IF (COUNT(PP)==0)
        [
        CALL CLRTOP(PP)
        WRITE(PP,10)
10      FORMAT('+WHICH BARREL LE=1 HE=2 ----------')
        CALL WRTBUF(PP)
        CALL MODSET(PP,1)
        COUNT(PP)=COUNT(PP)+1
        ]
    ELSE
        [
        N=COUNT(PP)
        GO TO (100,200,300,400),N
        COUNT(PP)=0
        RETURN
100     CALL GETBUF(PP,LINE)
        CALL RIGHTJ(LINE)
        DECODE(LINE,110)TEMP
110     FORMAT(F10.0)
        IF (TEMP^=1&TEMP^=2)
            [
            CALL CLRTOP(PP)
            WRITE(PP,120)
120         FORMAT('+ENTER 1 FOR LE OR 2 FOR HE -------')
            CALL WRTBUF(PP)
            ]
        ELSE
            [
            BAREL(PP)=TEMP
            IF(BAREL(PP)==1)
                BAR$(PP)='LE'
            ELSE
                BAR$(PP)='HE'
            CALL CLRTOP(PP)
            WRITE(PP,130)BAR$(PP)
130         FORMAT('+ENTER FINAL WEIGHT OF OLD ',A2,' BARREL ------')
            CALL WRTBUF(PP)
            COUNT(PP)=COUNT(PP)+1
            ]
        RETURN
```

```
200       BAR=BAREL(PP)
          CALL GETBUF(PP,LINE)
          CALL RIGHTJ(LINE)
          DECODE(LINE,110)FINWT(PP)
          CALL CLRTOP(PP)
          WRITE(PP,210)BAR$(PP)
210       FORMAT('+ENTER TOTAL SAMPLES FROM OLD ',A2,' BARREL -----')
          CALL WRTBUF(PP)
          COUNT(PP)=COUNT(PP)+1
          RETURN
300       BAR=BAREL(PP)
          CALL GETBUF(PP,LINE)
          CALL RIGHTJ(LINE)
          DECODE(LINE,110)SAMPLE(PP,BAR)
          ACCWT(PP,BAR)=ACCWT(PP,BAR)+FINWT(PP)-SAMPLE(PP,BAR)
          IF (PP==1)
              ACCWT(PP,BAR)=ACCWT(PP,BAR)-TARE(PP,BAR)
          CALL CLRTOP(PP)
          WRITE(PP,310)BAR$(PP)
310       FORMAT('+ENTER TARE OF NEW ',A2,' BARREL -------')
          CALL WRTBUF(PP)
          COUNT(PP)=COUNT(PP)+1
          RETURN
400       BAR=BAREL(PP)
          CALL GETBUF(PP,LINE)
          CALL RIGHTJ(LINE)
          DECODE(LINE,110)TARE(PP,BAR)
          CALL CLRTOP(PP)
          CALL WRTBUF(PP)
          COUNT(PP)=0
          CALL MODSET(PP,0)
          ]
      RETURN
      END
#
```

```
#
        SUBROUTINE CLRTOP(PP)
#
    LOGICAL HOME,CEOL
    INTEGER PP
#
#   THIS SUBROUTINE CLEARS THE TOP LINE OF THE
#   SPECIFIED TERMINAL
#
    CALL CLRBUF(PP)
    HOME=11
    CEOL=15
    Y=0
    X=0
    CALL CURSAD(PP,Y,X)
    WRITE(PP,10)HOME,CEOL
10  FORMAT('+',2A1)
    CALL WRTBUF(PP)
    RETURN
    END
#
    SUBROUTINE RIGHTJ(LINE)
#
    LOGICAL LINE(10)
#
#   FORTRAN INTERPRETS FOLLOWING SPACES ON A NUMBER
#   AS ZEROS.  SO THAT THIS DOES NOT CREATE A PROBLEM
#   ALL NUMBERS ENTERED DURING INTERRUPTS ARE RIGHT
#   JUSTIFIED BY THIS ROUTINE
#
    N=1
    REPEAT
        [
        IF(LINE(10)==SPACE)
            [
            FOR(I=10;I>=2;I=I-1)
                LINE(I)=LINE(I-1)
            LINE(1)=SPACE
            ]
        N=N+1
        ]UNTIL(LINE(10)^=SPACE\N>=10)
    RETURN
    END
#
```

```
#
      SUBROUTINE DAYIN(PP)
#
    LOGICAL LINE(10)
    INTEGER PP,J
    REAL TEMP
#
    INCLUDE COMMON.RFR
#
#   THIS INTERRUPT SUBROUTINE COLLECTS THE DAY OF THE YEAR
#   AND THE YEAR THEN SETS THE CLOCK ON THE FLUKE
#
    IF(COUNT(PP)==0)
        [
        CALL CLRBUF(PP)
        CALL CLRTOP(PP)
        WRITE(PP,10)
10      FORMAT('+ENTER DAY OF THE YEAR 1-366 ---------')
        CALL WRTBUF(PP)
        CALL MODSET(PP,1)
        COUNT(PP)=COUNT(PP)+1
        ]
    ELSE
        [
        IF(COUNT(PP)==1)
            [
            CALL GETBUF(PP,LINE)
            CALL RIGHTJ(LINE)
            DECODE(LINE,20)TEMP
20          FORMAT(F10.0)
            IF(TEMP<=0\TEMP>366)
                [
                CALL CLRBUF(PP)
                CALL CLRTOP(PP)
                WRITE(PP,30)
30              FORMAT('+NOT A VALID DAY TRY AGAIN---------')
                CALL WRTBUF(PP)
                ]
```

```
            ELSE IF(COUNT(PP)==2)
                [
                CALL GETBUF(PP,LINE)
                CALL RIGHTJ(LINE)
                DECODE(LINE,20)TEMP
                IF(TEMP<1980\TEMP>1990)
                    [
                    CALL CLRBUF(PP)
                    CALL CLRTOP(PP)
                    WRITE(PP,50)
50                  FORMAT('+NOT A VALID YEAR TRY AGAIN-------')
                    CALL WRTBUF(PP)
                    ]
                ELSE
                    [
                    YEAR=TEMP
                    CALL CLRBUF(PP)
                    CALL CLRTOP(PP)
                    CALL WRTBUF(PP)
                    COUNT(PP)=0
                    CALL MODSET(PP,0)
                    CALL SETCLK
                    CALL DUMPF
                    ]
                ]
        ]
    RETURN
    END
#
```

```
#
      SUBROUTINE WRTMAN(PP)
#
    LOGICAL LINE(10),BUF(60),NAME(11)
    INTEGER PP,J,IDAY,TH,DRIVE,N,FSTAT,FDAY,FHR,FRUN
    REAL TEMP,MANSTR(32)
#
    INCLUDE COMMON.RFR
#
    COMMON/DIR/FSTAT(128),FDAY(128),FHR(128),FRUN(128)
#
#   THIS INTERRUPT SUBROUTINE ASKS FOR THE DAY AND HOUR
#   WHEN THE MANUAL DATA WAS COLLECTED.  THEN IT ASKS THE
#   OPERATOR ALL THE QUESTIONS IN THE MANDAT?.DAT FILE.
#   THE ANSWER TO EACH QUESTION IS STORED AS A REAL NUMBER
#   IN THE 5TH SECTOR OF THE HOURLY RECORD.
#
    IF(COUNT(PP)==0)
        [
        FOR(J=1;J<=32;J=J+1)
            MANSTR(J)=0.0
        ENCODE(NAME,10)PP
10      FORMAT('MANDAT',I1,'.DAT')
        CALL OPEN (8,NAME,0)
        CALL CLRTOP(PP)
        WRITE(PP,20)
20      FORMAT('+ENTER DAY WHEN MANUAL DATA WAS COLLECTED----')
        CALL WRTBUF(PP)
        CALL MODSET(PP,1)
        COUNT(PP)=COUNT(PP)+1
        ]
    ELSE
        [
        IF(COUNT(PP)==1)
            [
            CALL GETBUF(PP,LINE)
            CALL RIGHTJ(LINE)
            DECODE(LINE,30)TEMP
30          FORMAT(F10.0)
            IF(TEMP<=0\TEMP>366)
                [
                CALL CLRTOP(PP)
                WRITE(PP,40)
40              FORMAT('+NOT A VALID DAY TRY AGAIN----')
                CALL WRTBUF(PP)
                ]
            ELSE
                [
                IDAY=TEMP
                CALL CLRTOP(PP)
                WRITE(PP,50)
50    FORMAT('+ENTER HOUR WHEN DATA WAS COLLECTED- 24=STARTUP,25=SHUTDOWN---
                CALL WRTBUF(PP)
                COUNT(PP)=COUNT(PP)+1
                ]
            ]
```

```
            ELSE IF(COUNT(PP)==2)
                 [
                 CALL GETBUF(PP,LINE)
                 CALL RIGHTJ(LINE)
                 DECODE(LINE,30)TEMP
                 IF(TEMP<0\TEMP>25)
                      [
                      CALL CLRTOP(PP)
                      WRITE(PP,60)
60                    FORMAT('+NOT A VALID HOUR TRY AGAIN----')
                      CALL WRTBUF(PP)
                      ]
                 ELSE
                      [
                      TH=TEMP
                      CALL CLRTOP(PP)
                      READ(8,70,ERR=500,END=500)BUF
70                    FORMAT(1X,60A1)
                      WRITE(PP,80)BUF
80                    FORMAT('+',60A1)
                      CALL WRTBUF(PP)
                      COUNT(PP)=COUNT(PP)+1
                      ]
                 ]
            ELSE
                 [
                 N=COUNT(PP)-2
                 CALL GETBUF(PP,LINE)
                 CALL RIGHTJ(LINE)
                 DECODE(LINE,30)MANSTR(N)
                 CALL CLRTOP(PP)
                 READ(8,70,END=600)BUF
                 WRITE(PP,80)BUF
                 CALL WRTBUF(PP)
                 COUNT(PP)=COUNT(PP)+1
                 ]
            ]
        RETURN
```

```
500 ENDFILE 8
    CALL CLRTOP(PP)
    WRITE(PP,90)PP
90  FORMAT('+NO MANDAT',I1,' FILE')
    CALL WRTBUF(PP)
    COUNT(PP)=0
    CALL MODSET(PP,0)
    RETURN
600 ENDFILE 8
    CALL CLRTOP(PP)
    DRIVE=PP+1
    ENCODE(NAME,100)PP
100 FORMAT('PLANT',I1,'  DAT')
    CALL OPEN(7,NAME,DRIVE)
    CALL GDIR
    FOR(N=1;(FDAY(N)^=IDAY\FHR(N)^=TH)&N<=128&FSTAT(N)^=0;N=N+1)
        ;
    IF(N==129\FSTAT(N)==0)
        [
        WRITE(PP,110)N
110     FORMAT('+N=',I4,'  DIRECTORY ERROR')
        ]
    ELSE
        [
        K=(N-1)*5+8+5
        WRITE(PP,115)N,K
115     FORMAT('+N=',I4,'  K=',I4)
        WRITE(7,REC=K)MANSTR
        FSTAT(N)=FSTAT(N)+1
        CALL WDIR
        ]
    ENDFILE 7
    CALL WRTBUF(PP)
    CALL MODSET(PP,0)
    COUNT(PP)=0
    RETURN
    END
#
```

```
#
      SUBROUTINE XCHNGD(PP)
#
    LOGICAL LINE(10),NAME(11)
    INTEGER PP,N,J,DRIVE,FSTAT,FDAY,FHR,FRUN,I,K,L,M
    REAL STORAR
#
    INCLUDE COMMON.RFR
#
    COMMON/STOR/STORAR(32,4)
#
    COMMON/DIR/FSTAT(128),FDAY(128),FHR(128),FRUN(128)
#
#   THIS INTERRUPT SUBROUTINE IS USED TO EXCHANGE DATA DISKS.
#   IF THE NEW DISK ALREADY HAS SOME DATA ON IT A WARNING MESSAGE
#   WILL BE PRINTED
#
    IF(COUNT(PP)==0)
        [
        CALL CLRTOP(PP)
        WRITE(PP,10)
10      FORMAT('+ARE YOU SURE ALL FILES ARE UP TO DATE----^Y=YES')
        CALL WRTBUF(PP)
        CALL MODSET(PP,2)
        COUNT(PP)=COUNT(PP)+1
        ]
    ELSE
        [
        IF(COUNT(PP)==1)
            [
            IF(ICR(PP)<0)
                [
                CALL GETBUF(PP,LINE)
                CALL CLRTOP(PP)
                WRITE(PP,20)
20              FORMAT('+USE ^U TO UNCOVER INCOMPLETE FILES')
                CALL WRTBUF(PP)
                CALL MODSET(PP,0)
                COUNT(PP)=0
                ]
            ELSE
                [
                CALL GETBUF(PP,LINE)
                CALL CLRTOP(PP)
                WRITE(PP,30)
30              FORMAT('+HAVE YOU CHANGED THE DISK----^Y=YES')
                CALL WRTBUF(PP)
                COUNT(PP)=COUNT(PP)+1
                ]
            ]
```

```
        ELSE IF(COUNT(PP)==2)
             [
             IF(ICR(PP)<0)
                  [
                  CALL GETBUF(PP,LINE)
                  CALL CLRTOP(PP)
                  WRITE(PP,40)
40                FORMAT('+WELL GO AHEAD AND CHANGE IT----^Y=DONE')
                  CALL WRTBUF(PP)
                  ]
             ELSE
                  [
                  CALL GETBUF(PP,LINE)
                  DRIVE=PP+1
                  CALL OPEN(8,'MARKER  DAT',DRIVE)
                  READ(8,50,ERR=100,END=100)LINE
50                FORMAT(10A1)
                  ENDFILE 8
                  CALL CLRTOP(PP)
                  WRITE(PP,60)
60                FORMAT('+NEW DISK IS NOT EMPTY')
                  GO TO 700
100               REWIND 8
                  WRITE(8,110)
110               FORMAT('+USED')
                  ENDFILE 8
                  FOR(N=1;N<=128;N=N+1)
                      FSTAT(N)=0
                  ENCODE(NAME,120)PP
120               FORMAT('PLANT',I1,'  DAT')
                  CALL OPEN(7,NAME,DRIVE)
                  CALL WDIR
                  ENDFILE 7
                  CALL CLRTOP(PP)
700               CALL WRTBUF(PP)
                  CALL MODSET(PP,0)
                  COUNT(PP)=0
                  ]
             ]
        ]
    RETURN
    END
#
```

```
#
        SUBROUTINE ZAP(PP)
#
    LOGICAL LINE(10)
    INTEGER PP
#
    INCLUDE COMMON.RFR
#
#   THIS INTERRUPT SUBROUTINE ALLOWS THE OPERATOR OF
#   EITHER PLANT TO STOP THE PROGRAM AND USE CDOS COMMANDS
#   WHEN THE OPERATOR THEN TYPES DISP CR ON THE PLANT 1
#   TERMINAL THE PROGRAM WILL RESTART UNDER THE SAME CONDITIONS
#   WHICH EXISTED WHEN IT WAS STOPPED
#
    IF(COUNT(PP)==0)
        [
        CALL CLRTOP(PP)
        WRITE(PP,10)
10      FORMAT('+DO YOU REALLY WANT TO ZAP THE PROGRAM---^Y=YES')
        CALL WRTBUF(PP)
        CALL MODSET(PP,2)
        COUNT(PP)=COUNT(PP)+1
        ]
    ELSE
        [
        IF(ICR(PP)<0)
            [
            CALL GETBUF(PP,LINE)
            CALL CLRTOP(PP)
            CALL WRTBUF(PP)
            CALL MODSET(PP,0)
            COUNT(PP)=0
            ]
        ELSE
            [
            CALL DUMPF
            CALL STOP
            ]
        ]
    RETURN
    END
#
```

```
#
      SUBROUTINE GDIR
#
     INTEGER FDIR, I, J, K, L, M
#
     COMMON/DIR/FDIR(512)
#
#    THIS SUBROUTINE READS THE FIRST 1K BLOCK
#    OF THE DATA FILE WHICH CONTAINS THE DIRECTORY
#
     FOR(I=0; I<=7; I=I+1)
          [
          J=I+1
          K=I*64+1
          L=K+63
          READ(7, REC=J)(FDIR(M), M=K, L)
          ]
     RETURN
     END
#
      SUBROUTINE WDIR
#
     INTEGER FDIR, I, J, K, L, M
#
     COMMON/DIR/FDIR(512)
#
#    THIS SUBROUTINE WRITES THE UP DATED DIRECTORY
#    TO THE FIRST K OF THE DATA FILE
#
     FOR(I=0; I<=7; I=I+1)
          [
          J=I+1
          K=I*64+1
          L=K+63
          WRITE(7, REC=J)(FDIR(M), M=K, L)
          ]
     RETURN
     END
#
```

```
#
        SUBROUTINE EDIT(PP,MODE)
#
    LOGICAL LINE(10)
    INTEGER MODE,PP,CHN
    REAL TEMP
#
    INCLUDE COMMON.RFR
#
#   THIS INTERRUPT SUBROUTINE ALLOWS THE OPERATOR TO
#   EDIT EITHER THE CENTER POINT OR TOLERANCE FOR ANY
#   CHANNEL OR CALCULATION ASSOCIATED WITH HIS PLANT
#   TO EDIT THE CENTER POINT USE ^E
#   TO EDIT THE TOLERANCE USE ^T
#
    IF(COUNT(PP)==0)
        [
        CALL CLRTOP(PP)
        WRITE(PP,10)
10      FORMAT('+ENTER CHANNEL NUMBER TO BE EDITED----')
        CALL WRTBUF(PP)
        CALL MODSET(PP,1)
        COUNT(PP)=COUNT(PP)+1
        ]
    ELSE
        [
        IF(COUNT(PP)==1)
            [
            CALL GETBUF(PP,LINE)
            CALL RIGHTJ(LINE)
            DECODE(LINE,20)TEMP
20          FORMAT(F10.0)
            IF(TEMP<0\TEMP>=223)
                [
                CALL CLRTOP(PP)
                WRITE(PP,30)
30              FORMAT('+NOT A VALID CHANNEL TRY AGAIN----')
                CALL WRTBUF(PP)
                ]
```

```
            ELSE
                [
                CHN=TEMP
                IF(PLANT(CHN+1)^=PP)
                    [
                    CALL CLRTOP(PP)
                    WRITE(PP,40)CHN
40                  FORMAT('+CHN ',I3,' NOT ON YOUR PLANT TRY AGAIN---')
                    CALL WRTBUF(PP)
                    ]
                ELSE
                    [
                    CALL CLRTOP(PP)
                    IF(MODE==1)
                        [
                        WRITE(PP,50)CHN,CENTER(CHN+1)
50  FORMAT('+CHN.  ',I3,' CENTER=',F10.3,' ENTER NEW VALUE---')
                        ]
                    ELSE
                        [
                        WRITE(PP,60)CHN,SPAN(CHN+1)
60  FORMAT('+CHN.  ',I3,' TOLERANCE=',F10.3,' ENTER NEW VALUE----')
                        ]
                    CALL WRTBUF(PP)
                    COUNT(PP)=COUNT(PP)+1
                    ]
                ]
            ]
        ELSE
            [
            CALL GETBUF(PP,LINE)
            CALL RIGHTJ(LINE)
            DECODE(LINE,20)TEMP
            IF(MODE==1)
                CENTER(CHN+1)=TEMP
            ELSE
                SPAN(CHN+1)=TEMP
            CALL CLRTOP(PP)
            CALL WRTBUF(PP)
            CALL MODSET(PP,0)
            COUNT(PP)=0
            CALL FILL(PP,PG(PP))
            CALL DUMPF
            ]
        ]
    RETURN
    END
#
```

```
#
      SUBROUTINE UNCOV(PP)
#
      LOGICAL HON,HOFF,BELL,LINE(10),NAME(11)
      INTEGER OLDPG(2),DRIVE,PP,N,FSTAT,FDAY,FHR,FRUN
#
      INCLUDE COMMON.RFR
#
      COMMON/DIR/FSTAT(128),FDAY(128),FHR(128),FRUN(128)
#
#     THIS INTERRUPT SUBROUTINE WRITES THE DIRECTORY TO THE
#     SCREEN IN THE FOLLOWING FORMAT - RUN DAY HR STATUS -
#     ANY INCOMPLETE FILES ARE WRITTEN AT FULL INTENSITY
#     WHILE FILES WHICH HAVE HAD THE MANUAL DATA WRITTEN TO
#     THEM ARE WRITTEN AT HALF INTENSITY.
#     A MAXIMUM OF 120 FILES CAN BE SHOWN ON ONE PAGE.  THIS REPRESENTS
#     5 DAYS OF DATA.  IF MORE THAN 120 FILES ARE PRESENT (MAX=128) THE
#     FIRST FEW FILES WILL BE OVERWRITTEN BY THE LAST FEW.  IF THIS HAPPENS
#     A WARNING MESSAGE WILL ALSO BE WRITTEN TELLING THE OPERATOR TO CHANGE
#     DISKS.
#
```

```
    IF(COUNT(PP)==0)
        [
        CALL CLEAR(PP)
        CALL CLRTOP(PP)
        OLDPG(PP)=PG(PP)
        PG(PP)=10
        DRIVE=PP+1
        ENCODE(NAME,10)PP
10      FORMAT('PLANT',I1,'  DAT')
        CALL OPEN(7,NAME,DRIVE)
        CALL GDIR
        ENDFILE 7
        HON=2
        WRITE(PP,12)HON
12      FORMAT('+',A1)
        FOR(N=1;FSTAT(N)^=0&N<=128;N=N+1)
            [
            IF(FSTAT(N)==1)
                [
                HOFF=4
                HON=2
                ]
            ELSE
                [
                HOFF=27
                HON=27
                ]
            WRITE(PP,20)HOFF,FRUN(N),FDAY(N),FHR(N),FSTAT(N),HON
20          FORMAT('+',A1,2I4,I3,I2,A1,'\',2X)
            ]
        HOFF=4
        WRITE(PP,12)HOFF
        BELL=7
        IF(N>=120)
            [
            WRITE(PP,40)BELL,BELL,BELL,BELL,BELL
40          FORMAT('+DIRECTORY OVERLOAD CHANGE DISK',5A1)
            ]
        ELSE IF(N==1)
            [
            WRITE(PP,50)
50          FORMAT('+DIRECTORY EMPTY HIT RETURN')
            ]
        CALL WRTBUF(PP)
        CALL MODSET(PP,1)
        COUNT(PP)=COUNT(PP)+1
        ]
    ELSE
        [
        CALL GETBUF(PP,LINE)
        CALL CLRTOP(PP)
        CALL WRTBUF(PP)
        CALL MODSET(PP,0)
        COUNT(PP)=0
        PG(PP)=OLDPG(PP)
        CALL FILL(PP,PG(PP))
        ]
    RETURN
    END
#
```

```
#
      SUBROUTINE STRTUP(PP)
#
    LOGICAL LINE(10)
    INTEGER PP, J, CHN1, CHN2, CHN3, SW, BAR
    REAL TEMP, TM
#
    INCLUDE COMMON.RFR
#
#   THIS INTERRUPT SUBROUTINE IS USED TO CHANGE THE PLANT MODE
#   FROM SHUTDOWN TO RUNNING.  IF A RUN IS ALREADY IN PROGRESS
#   THIS PROGRAM WILL BE UNACCESSABLE TO THE OPERATOR.
#
    IF(COUNT(PP)==0)
        [
        CALL CLRTOP(PP)
        WRITE(PP,10)
10      FORMAT('+ENTER 8 CHARACTER DESCRIPTION OF RUN---')
        CALL MODSET(PP,3)
        CALL WRTBUF(PP)
        COUNT(PP)=COUNT(PP)+1
        ]
    ELSE
        [
        J=COUNT(PP)
        GO TO (100,200,300,400),J
        COUNT(PP)=0
        RETURN
100     CALL GETBUF(PP,LINE)
        DECODE(LINE,110)RUN(PP)
110     FORMAT(A8)
        CALL MODSET(PP,1)
        CALL CLRTOP(PP)
        WRITE(PP,120)
120     FORMAT('+ENTER RUN NUMBER 1-999 ----')
        CALL WRTBUF(PP)
        COUNT(PP)=COUNT(PP)+1
        RETURN
200     CALL GETBUF(PP,LINE)
        CALL RIGHTJ(LINE)
        DECODE(LINE,210)TEMP
210     FORMAT(F10.0)
        CALL CLRTOP(PP)
        IF(TEMP<1\TEMP>999)
            [
            WRITE(PP,220)
220         FORMAT('+NOT A VALID RUN NUMBER TRY AGAIN 1-999 ---')
            ]
        ELSE
            [
            NRUN(PP)=TEMP
            WRITE(PP,230)
230         FORMAT('+ENTER LE BARREL TARE---')
            COUNT(PP)=COUNT(PP)+1
            ]
        CALL WRTBUF(PP)
        RETURN
```

```
300       CALL GETBUF(PP,LINE)
          CALL RIGHTJ(LINE)
          DECODE(LINE,210)TARE(PP,1)
          CALL CLRTOP(PP)
          WRITE(PP,310)
310       FORMAT('+ENTER HE BARREL TARE---')
          COUNT(PP)=COUNT(PP)+1
          CALL WRTBUF(PP)
          RETURN
400       CALL GETBUF(PP,LINE)
          CALL RIGHTJ(LINE)
          DECODE(LINE,210)TARE(PP,2)
          CHN1=FEDLOC(1,PP)
          CHN2=FEDLOC(2,PP)
          CHN3=FEDLOC(3,PP)
          INITWT(PP,1)=VAL(CHN2+1)
          INITWT(PP,2)=VAL(CHN3+1)
          IF(VAL(CHN1+1)>1.5)
               SW=1
          ELSE
               SW=2
          OLDSW(PP)=SW
          OLDWT(PP)=INITWT(PP,SW)
          WTSREM(PP)=0
          FEDTOT(PP)=0.
          FOR(BAR=1;BAR<=2;BAR=BAR+1)
               [
               SAMPLE(PP,BAR)=0.
               ACCWT(PP,BAR)=0.
               BARWT(PP,BAR)=0.
               ]
          TM=HR*3600+MIN*60+SEC
          FOR(I=1;I<=5;I=I+1)
               [
               TOT(PP,I)=0.
               TMSEC(PP,I)=TM
               ]
          CALL HRLY(PP,24)
          STAT(PP)=1
          CALL DUMPF
          CALL CLRTOP(PP)
          CALL WRTBUF(PP)
          COUNT(PP)=0
          CALL MODSET(PP,0)
          ]
     RETURN
     END
#
```

```
#
      SUBROUTINE SHTDWN(PP)
#
    LOGICAL LINE(10)
    INTEGER PP,J
#
    INCLUDE COMMON.RFR
#
#   THIS INTERRUPT SUBROUTINE IS USED TO SHUTDOWN A RUN
#   IF THE PLANT IS ALREADY SHUTDOWN THIS PROGRAM WILL NOT
#   RESPOND.
#
    IF(COUNT(PP)==0)
        [
        CALL CLRTOP(PP)
        WRITE(PP,10)
10      FORMAT('+ENTER FINAL LE SAMPLES---')
        CALL MODSET(PP,1)
        CALL WRTBUF(PP)
        COUNT(PP)=COUNT(PP)+1
        ]
    ELSE
        [
        J=COUNT(PP)
        GO TO (100,200),J
        COUNT(PP)=0
        RETURN
100     CALL GETBUF(PP,LINE)
        CALL RIGHTJ(LINE)
        DECODE(LINE,210)SAMPLE(PP,1)
        CALL CLRTOP(PP)
        WRITE(PP,120)
120     FORMAT('+ENTER FINAL HE SAMPLE----')
        CALL WRTBUF(PP)
        COUNT(PP)=COUNT(PP)+1
        RETURN
```

```
200       CALL GETBUF(PP,LINE)
          CALL RIGHTJ(LINE)
          DECODE(LINE,210)SAMPLE(PP,2)
210       FORMAT(F10.0)
          CALL HRLY(PP,25)
          ENCODE(LINE,220)
220       FORMAT('SHUTDOWN  ')
          DECODE(LINE,230)RUN(PP)
230       FORMAT(A8)
          NRUN(PP)=0
          STAT(PP)=0
          CALL DUMPF
          CALL CLRTOP(PP)
          CALL WRTBUF(PP)
          COUNT(PP)=0
          CALL MODSET(PP,0)
          ]
      RETURN
      END
```

A.11

ONLINE.RFR

```
        SUBROUTINE ONLINE(CHN)
#
      INTEGER CHN,CALC
#
#     IF THE CHANNEL NUMBER IS GREATER THAN 179
#     IT IS ASSUMED TO BE A CALCULATION AND CONTROL
#     IS PASSED TO THIS PROGRAM WHICH DISPATCHES THE PROGRAM
#     TO THE APPROPRIATE CALCULATION ROUTINE
#
      CALC=CHN-179
      GO TO (100,200,200,200,200,300,300,300,300,400),CALC
      IF(CHN<202)
          GOTO 500
      CALC=CHN-201
      GO TO (100,200,200,200,200,300,300,300,300,400),CALC
      IF(CHN<224)
          GOTO 500
      RETURN
100   CALL FEED(CHN)
      RETURN
200   CALL HTC(CHN)
      RETURN
300   CALL DTRC(CHN)
      RETURN
400   CALL GASR(CHN)
      RETURN
500   RETURN
      END
#
```

```
#
      SUBROUTINE FEED(CHN)
    INTEGER PP, I, N, SW, NOTSW, CHN, CHN1, CHN2, CHN3
    REAL TM, TEMTOT, TANKWT(2)
#
    INCLUDE COMMON.RFR
#
#   THIS SUBROUTINE CALCULATES THE FEED RATE
#
    PP=PLANT(CHN+1)
    CHN1=FEDLOC(1,PP)
    CHN2=FEDLOC(2,PP)
    CHN3=FEDLOC(3,PP)
    CALL AVCHN(CHN1)
    CALL DISP(CHN1)
    CALL AVCHN(CHN2)
    CALL DISP(CHN2)
    CALL AVCHN(CHN3)
    CALL DISP(CHN3)
    TANKWT(1)=VAL(CHN2+1)
    TANKWT(2)=VAL(CHN3+1)
    IF (VAL(CHN1+1)>1.5)
        [
        SW=1
        NOTSW=2
        ]
    ELSE
        [
        SW=2
        NOTSW=1
        ]
```

```
IF (SW^=OLDSW(PP))
     [
     FEDCUR=INITWT(PP,NOTSW)-TANKWT(NOTSW)+WTSREM(PP)*10
     FEDTOT(PP)=FEDTOT(PP)+FEDCUR
     WTSREM(PP)=0
     CALL DUMPF
     ]
ELSE IF (TANKWT(SW)>OLDWT(PP)+5)
     [
     WTSREM(PP)=WTSREM(PP)+1
     CALL DUMPF
     ]
FEDCUR=INITWT(PP,SW)-TANKWT(SW)+WTSREM(PP)*10
TEMTOT=FEDTOT(PP)+FEDCUR
TM=HR*3600+MIN*60+SEC
FOR (I=1;I<=4;I=I+1)
     [
     N=I+1
     TOT(PP,I)=TOT(PP,N)
     TMSEC(PP,I)=TMSEC(PP,N)
     ]
TOT(PP,5)=TEMTOT
TMSEC(PP,5)=TM
TM=TMSEC(PP,5)-TMSEC(PP,1)
TM=TM/3600
IF (TM<=0)
     TM=TM+24
VAL(CHN+1)=(TOT(PP,5)-TOT(PP,1))/TM
CALL DISP(CHN)
OLDSW(PP)=SW
OLDWT(PP)=TANKWT(SW)
INITWT(PP,NOTSW)=TANKWT(NOTSW)
RETURN
END
#
```

```
#
        SUBROUTINE HTC(CHN)
#
    INTEGER PP, I, N, CHN, TEMCHN
    REAL TEMP(4)
#
    INCLUDE COMMON.RFR
#
#   THIS SUBROUTINE CALCULATES THE HEAT TRANSFER CO-EFFICIENT
#
    IF (CHN<203)
        N=(CHN-181)*4+1
    ELSE
        N=(CHN-203)*4+17
    K=N+3
    J=1
    FOR (I=N; I<=K; I=I+1)
        [
        TEMCHN=HTCLOC(I)
        CALL AVCHN(TEMCHN)
        TEMP(J)=VAL(TEMCHN+1)
        J=J+1
        CALL DISP(TEMCHN)
        ]
    VAL(CHN+1)=CONST(1)*(TEMP(1)-(TEMP(2)+TEMP(3)/2))/(TEMP(4)-TEMP(1))
    CALL DISP(CHN)
    RETURN
    END
#
```

```
#
      SUBROUTINE DTRC(CHN)
#
    INTEGER CHN, TEMCHN
    REAL TEMP(3)
#
    INCLUDE COMMON.RFR
#
#   THIS SUBROUTINE CALCULATES TEMPERATURE DIFFERENCES
#
    IF(CHN<202)
        N=(CHN-185)*3+1
    ELSE
        N=(CHN-207)*3+1
    K=N+3
    J=1
    FOR (I=N; I<=K; I=I+1)
        [
        CALL AVCHN(DIFLOC(I))
        TEMCHN=DIFLOC(I)
        TEMP(J)=VAL(TEMCHN+1)
        J=J+1
        CALL DISP(DIFLOC(I))
        ]
    VAL(CHN+1)=(TEMP(1)+TEMP(2))/2-TEMP(3)
    CALL DISP(CHN)
    RETURN
    END
#
```

```
#
      SUBROUTINE GASR(CHN)
#
      INTEGER PP, CHN, TEMLOC
      REAL GRPD, H2P, MOLEPC, ATEMP
#
      INCLUDE COMMON.RFR
#
#     THIS SUBROUTINE CALCULATES THE GAS RATE
#
      PP=PLANT(CHN+1)
      TEMLOC=GASLOC(1,PP)
      CALL AVCHN(TEMLOC)
      CALL DISP(TEMLOC)
      GRPD=VAL(TEMLOC+1)
      TEMLOC=GASLOC(2,PP)
      CALL AVCHN(TEMLOC)
      CALL DISP(TEMLOC)
      H2P=VAL(TEMLOC+1)
      TEMLOC=GASLOC(3,PP)
      CALL AVCHN(TEMLOC)
      CALL DISP(TEMLOC)
      MOLEPC=VAL(TEMLOC+1)
      TEMLOC=GASLOC(4,PP)
      CALL AVCHN(TEMLOC)
      CALL DISP(TEMLOC)
      ATEMP=VAL(TEMLOC+1)
      VAL(CHN+1)=CONST(2)*SQRT(GRPD)/(CONST(3)-CONST(4)*MOLEPC)*SQRT(H2P)
      VAL(CHN+1)=VAL(CHN+1)*287.0/(273.0+ATEMP)
      CALL DISP(CHN)
      RETURN
      END
```

A.12

COMPAC.RFR

```
      DEFINE( FLUKE, 4)
      DEFINE( SPACE, 32)
      DEFINE( CHRA, 65)
      DEFINE( CHRY, 89)
      DEFINE( CHRD, 68)
      DEFINE( ENDOFSCAN, 20)
      DEFINE( ESC, 27)
      DEFINE( COMMAT, 64)
      DEFINE( LF, 10)
      DEFINE( NUMSNG, 35)
      DEFINE( STAR, 42)
#
          PROGRAM COMPAC
#
      INCLUDE COMMON.RFR
#
#     THIS PROGRAM CALLS THE NECESSARY SUBROUTINES
#     TO COMPRESS THE TABLE DATA TO A FORM THAT
#     THE MACHINE CAN USE MORE EFFICIENTLY
#
      CALL INITAR
      CALL LDBIG
      CALL LDBATO
      CALL LDLOCS
      CALL LDKSTS
      CALL DUMP
      END
#
#
```

```
      SUBROUTINE LDBIG
#
#   THIS SUBROUTINE LOADS THE DATA FROM BIGTAB.DAT
#
    LOGICAL LINE(85),UNIT(8),F(11)
    INTEGER I,N,K,PP
#
    INCLUDE COMMON.RFR
#
        CALL OPEN (7,'BIGTAB  DAT',0)
        REPEAT
            [
            CALL GETLN(LINE)
            IF(LINE(1)==STAR)
                BREAK
            FOR(I=15;I<84;I=I+1)
                LINE(I)=LINE(I+2)
            DECODE (LINE,42)I
42          FORMAT (I3)
            N=I+1
            DECODE (LINE,44)PLANT(N),DESC(N),PRIOR(N),AVEN(N),PAGE(N),X(N),
            Y(N),SECT(N),WORD(N),A(N),B(N),CENTER(N),SPAN(N)
44          FORMAT (3X,I3,2X,A8,7I3,4F10.0)
            ]
60      ENDFILE 7
        RETURN
        END
#
```

```
      SUBROUTINE LDBATO
#
      INTEGER SHORTC,LONGC
#
      INCLUDE COMMON.RFR
#
#     THIS SUBROUTINE READS THE BATTING ORDER FROM
#     FILES SHORT.DAT AND LONG.DAT
#
      CALL OPEN(7,'SHORT    DAT',0)
      SHORTC=1
      REPEAT
          [
          READ(7,10,END=100)SHORT(SHORTC)
10        FORMAT(I3)
          SHORTC=SHORTC+1
          ]UNTIL(SHORTC>100)
      WRITE(3,20)
20    FORMAT(' SHORT FILE TOO LONG')
100   ENDFILE 7
      CALL OPEN(7,'LONG     DAT',0)
      LONGC=1
      REPEAT
          [
          READ(7,10,END=200)LONG(LONGC)
          LONGC=LONGC+1
          ]UNTIL(LONGC>500)
      WRITE(3,30)
30    FORMAT(' LONG FILE TOO LONG')
200   ENDFILE 7
      RETURN
      END
#
```

```
      SUBROUTINE LDKSTS
#
      INTEGER KSTC
#
      INCLUDE COMMON.RFR
#
#     THIS SUBROUTINE READS THE LIST OF CONSTANTS
#     USED IN THE ONLINE CALCULATIONS FROM THE
#     FILE KSTS.DAT
#
      CALL OPEN(7,'KSTS    DAT',0)
      KSTC=1
      REPEAT
          [
          READ(7,10,END=100)CONST(KSTC)
10        FORMAT(F10.0)
          KSTC=KSTC+1
          ]UNTIL(KSTC>10)
      WRITE(3,20)
20    FORMAT(' KSTS FILE TOO LONG')
100   ENDFILE 7
      RETURN
      END
#
```

```
          SUBROUTINE DUMP
#
#
     INCLUDE COMBLOK.RFR
#
#    THIS SUBROUTINE DUMPS A FILE NAMED
#    DISP.DAT TO THE CURRENT DISK.  THIS
#    FILE CONTAINS ALL THE DATA FROM THE TABLES
#    BUT IN AN UNFORMATTED BLOCK
#
     CALL OPEN(7,'DISP    DAT',0)
     FOR (I=0;I<=65;I=I+1)
          [
          J=I+1
          K=I*32+1
          L=K+31
          WRITE(7,REC=J)(BLOCK(M),M=K,L)
          ]
     WRITE(7,REC=67)(BLOCK(M),M=2113,2137)
     FOR (I=0;I<=21;I=I+1)
          [
          J=I+68
          K=I*64+1
          L=K+63
          WRITE(7,REC=J)(FBLOK(M),M=K,L)
          ]
     WRITE(7,REC=90)(FBLOK(M),M=1409,1451)
     ENDFILE 7
     RETURN
     END
#
#
```

```
      SUBROUTINE LDLOCS
#
      LOGICAL LINE(85)
#
      INCLUDE COMMON.RFR
#
#     THIS SUBROUTINE READS FILE LOCS.DAT
#     THIS FILE CONTAINS THE CHANNEL NUMBERS
#     CORESPONDING TO THE VARIABLES IN THE
#     ONLINE CALCULATIONS
#
      CALL OPEN(7,'LOCS    DAT',0)
      FOR(I=1;I<=2;I=I+1)
          [
          CALL GETLN(LINE)
          DECODE(LINE,40)(FEDLOC(K,I),K=1,3)
40        FORMAT(3I4)
          ]
      FOR(I=0;I<=7;I=I+1)
          [
          CALL GETLN(LINE)
          J=I*4+1
          L=J+3
          DECODE(LINE,50)(HTCLOC(K),K=J,L)
50        FORMAT(4I4)
          ]
      FOR(I=0;I<=7;I=I+1)
          [
          CALL GETLN(LINE)
          J=I*3+1
          L=J+2
          DECODE(LINE,60)(DIFLOC(K),K=J,L)
60        FORMAT(3I4)
          ]
      FOR(I=1;I<=2;I=I+1)
          [
          CALL GETLN(LINE)
          DECODE(LINE,70)(GASLOC(K,I),K=1,4)
70        FORMAT(4I4)
          ]
      ENDFILE 7
      RETURN
      END
#
```

```
          SUBROUTINE INITAR
#
      INCLUDE COMMON.RFR
#
#     THIS SUBROUTINE INITIALIZES ALL THE ARRAYS SO THAT
#     EVEN IF NO DATA FOR A VARIBLE IS READ THAT VARIABLE
#     WILL NOT BE UNDEFINED
#
      FOR(I=1;I<=224;I=I+1)
          [
          PLANT(I)=0
          PAGE(I)=3
          X(I)=80
          Y(I)=24
          AVEN(I)=1
          SECT(I)=1
          WORD(I)=1
          PRIOR(I)=0
          A(I)=1
          B(I)=0
          DESC(I)='NOTVALID'
          VAL(I)=0
          CENTER(I)=500
          SPAN(I)=-500
          ]
      FOR(I=1;I<=500;I=I+1)
          LONG(I)=500
      FOR(I=1;I<=100;I=I+1)
          SHORT(I)=500
      FOR(I=1;I<=10;I=I+1)
          CONST(I)=1
      ENCODE(RUN,100)
  100 FORMAT('SHUTDOWNSHUTDOWN')
      RETURN
      END
#
      BLOCK DATA INDAT
#
      INCLUDE COMMON.RFR
#
#     THIS BLOCK DATA SUBROUTINE INITIALIZES THE NON-CHANNEL
#     DATA
#
      DATA INITWT/4*0.0/,WTSREM/0,0/,OLDSW/0,0/,OLDWT/0.0,0.0/,
          FEDTOT/0.0,0.0/,TOT/10*0.0/,TMSEC/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./,
          TARE,SAMPLE,ACCWT,BARWT/14*0.0/,DAY,HR,MIN,SEC,YEAR,
          OLDHR/1,0,0,0,1980,12/,STAT/0,0/,NRUN/2*0/
#
      END
#
```

```
        SUBROUTINE GETLN(LINE)
#
    LOGICAL LINE(85)
#
#   THIS SUBROUTINE READS A LINE FROM THE TABLES
#   AND REMOVES ANY LINE FEED CHARACTERS FROM
#   THE BEGINNING.  IT ALSO SKIPS LINES BEGINNING
#   WITH A '#' CHARACTER.  IF AN END OF FILE IS FOUND
#   A STAR (*) IS RETURNED.
#
    REPEAT
        [
        READ (7,40,END=60)LINE
40      FORMAT (85A1)
        IF(LINE(1)==LF)
            [
            FOR(I=1; I<85; I=I+1)
                LINE(I)=LINE(I+1)
            ]
        ]UNTIL (LINE(1)^=NUMSNG)
    RETURN
60  LINE(1)=STAR
    RETURN
    END
```

POOR
QUALITY
ORIGINAL